Computational and Symbolic Proofs of Security: a short report

Mario Strefler[1], Chunhua Su[2], Vorapong Suppakitpaisarn[3], Itsuki Suzuki[4], and Andrea Turrini[5]

[1] Universität Karlsruhe, Germany
[2] Kyushu University, Japan
[3] The University of Tokyo, Japan
[4] The University of Osaka, Japan
[5] University of Verona, Italy

Foreword

The Spring School and Workshop on Computational and Symbolic Proofs of Security (CoSyProofs 2009) took place in Izu-Atagawa, from April 5, 2009 until April 9, 2009. It was a joint meeting with the French-Japanese collaboration project. It was sponsored By

– NTT (Nippon Telephone and Telegraph Communications),
– JST (Japan Science Technology) & CNRS (Centre National de la Recherche Scientifique), through the French-Japanese collaboration project *Computational and Symbolic Proofs of Security*
– AIST (Advanced Industrial Science and Technology)

It gathered 62 researchers. The participants mainly came from Japan (32) and France (16), but also from other countries (14). There were 5 tutorials (by M. Abadi & B. Warinschi, M.Backes & M. Berg, J. Mitchell, O. Pereira & R. Küsters, R. Segala), 3 invited talks (by D. Pointcheval, B. Blanchet, K. Ohta) and 12 technical presentations.

This document is a short report on some of the presentations, that was written by 5 students, who participated in the meeting.

Hubert Comon-Lundh

1 Introduction to Computational Soundness

Speaker: Martín Abadi, University of California at Santa Cruz

1.1 Cryptography and Computational Soundness

Two distinct, rigorous views of cryptography have developed over the years, in two mostly separate communities. One of the views relies on a simple but effective formal approach; the other, on a detailed computational model that considers issues of complexity and probability. There is an uncomfortable and interesting gap between these two approaches to cryptography. This paper starts to bridge the gap, by providing a computational justification for a formal treatment of encryption. The goal is to get the soundness of formal analysis and to prove that axioms assumed in the formal model are true for some cryptographic construction. Formal proofs must be sound in following sense: For any attack in the concrete (computational) model, there exists a matching attack in the abstract (formal) model, or else the concrete attack violates computational security of some cryptographic primitive. If we do not find an attack in the formal model, then no computational attack exists. More precisely, probability that a computational attack exists is negligible.

From the formal point of view, there is a large body of literature that treats cryptographic operations in a purely formal way [1]. For example, the expression $\{M\}_K$ may represent an encrypted message, with plaintext M and key K. All $\{M\}_K$, M, and K are formal expressions, rather than sequences of bits. Various functions can be applied to such expressions, yielding other expressions. One of them is decryption, which produces M from $\{M\}_K$ and K. Crucially, it is supposed that there is no way to recover M or K from $\{M\}_K$ alone. Thus, the idealized security properties of encryption are modeled (rather than defined). They are built into the model of computation on expressions.

Soundness property (desired): If a security property can be proved formally, then it holds in the computational model. The formal proof will

– not mention probabilities and complexities,
– consider attacks only in the formal model, and
– establish an all-or-nothing statement.

Soundness means that the statement is true with high probability for all computationally reasonable attacks.

1.2 Formal Encryption and Expression Equivalence

The speaker gives out the formal view of cryptography, specifically treating symmetric encryption. He also describes the space of expressions on which encryption operates, and what it means for two expressions to be equivalent. Equivalence relations are useful in semantics of modal logics: in such semantics, one says that two states of a computation ”look the same” to a principal only if the principal has equivalent expressions in those states. Equivalence relations also appear in bisimulation proof techniques, where one requires that bisimilar processes produce equivalent messages.

We write *Bool* for the set of bits $\{0,1\}$. These bits can be used to spell out numbers and principal names, for example. We write *Keys* for a fixed, nonempty set of symbols disjoint from *Bool*. The symbols $K, K', K'', \ldots$ and $K_1, K_2, \ldots$ are all in *Keys*. Informally, elements of the set *Keys* represent cryptographic keys, generated randomly by a principal that is constructing an expression. Formally, however, keys are atomic symbols, not strings of bits.

$M, N ::=$	expressions
i	bits (for $i \in \mathbf{Bool}$)
K	keys (for $K \in \mathbf{Keys}$)
(M, N)	pairs
$\{M\}_K$	symmetric encryptions

Fig. 1. The set of expressions *Exp*

There are several possible extensions of the set of expressions:

– We could allow expressions of the form $\{M\}_N$, where an arbitrary expression N is used as encryption key.
– We could distinguish encryption keys from decryption keys, as in public-key cryptosystems.

1.3 The Computational Soundness of Formal Equivalence

Informally, two expressions are equivalent if they look the same to an attacker. Formally, two expressions are equivalent if they yield the same pattern (up to renaming). For example,

– $0 \cong 0$
– $0 \ncong 1$
– $\{0\}_K \cong \{1\}_K$
– $(K, \{0\}_K) \ncong (K, \{1\}_K)$
– $(K, \{\{0\}_{K'}\}_K \cong (K, \{\{1\}_{K'}\}_K)$
– $(\{0\}_K, \{0\}_K) \cong (\{0\}_K, \{1\}_K)$

This property justifies equivalences such as the one above, where the two plaintexts are of different sizes. In an implementation, it can be guaranteed by padding plaintexts up to a maximum size, and truncating larger expressions or mapping them to some fixed string.

1.4 Adaptive Security and Multicast

So far, we were concerned with the equivalence of two expressions M and N. More generally, we may consider sequences of expressions (such as $M_0, M_1, \ldots$ and $N_0, N_1, \ldots$). The adversary produces the sequences. They are evaluated under a fixed key assignment. This is a step towards general active adversaries. The sequences may be produced adaptively: each expression may depend on previous interactions. We need to assume that these expressions do not contain cycles and do not reveal previously secret keys. Under this assumption, the soundness theorem says:

1. The adversary picks two equivalent sequences.
2. Then the adversary cannot distinguish the bitstrings that correspond to the two sequences, computationally.

An extension applies when some expressions can represent pseudorandom generators ($G_i(K)$).

1.5 Cryptographic Access Control for XML

This method was proposed in Abadi et al. [2]. As a formal counterpart to their loose, informal concept of data secrecy, we introduce a strong, precise cryptographic definition. The definition goes roughly as follows. Consider a protection for an XML document. An adversary is given an arbitrary set of keys, and the liberty of selecting two instantiations for the data in all nodes that occur in the XML document. The only restriction on these instantiations is that they should coincide on the nodes to which the adversary rightfully has access according to its keys and the abstract semantics of protections. In other words, the adversary selects two documents that contain the same information in the nodes it can access but may differ elsewhere. Then the adversary is given the partially encrypted document that corresponds to one of the two documents, and its goal is to decide which of the two instantiations was used in generating this partially encrypted document. Security means that the adversary cannot do much better than picking the document at random. This implies that the partially encrypted document reveals no information on the data in the nodes that should be hidden to the adversary, otherwise this information would be sufficient to determine which instantiation was used.

A protection is an XML tree which nodes are guarded by positive boolean formulas over a set of symbols $K_1, K_2, \ldots$ that stand for cryptographic keys.

Using simple transformations, one can rewrite any protection into an equivalent, normalized protection where all formulas that guard nodes are atomic, that is, one of *true*, *false*, or K for some $K \in \mathit{Keys}$. Normalization requires adding metadata nodes, keys, and key shares. Normalization can also include removing parts guarded by *false*, so we assume that *false* does not appear in normalized formulas (departing slightly from the original definition but without loss of generality). Normalized protections are important since, in standard encryption schemes, one can encrypt under an atomic key but not under a boolean

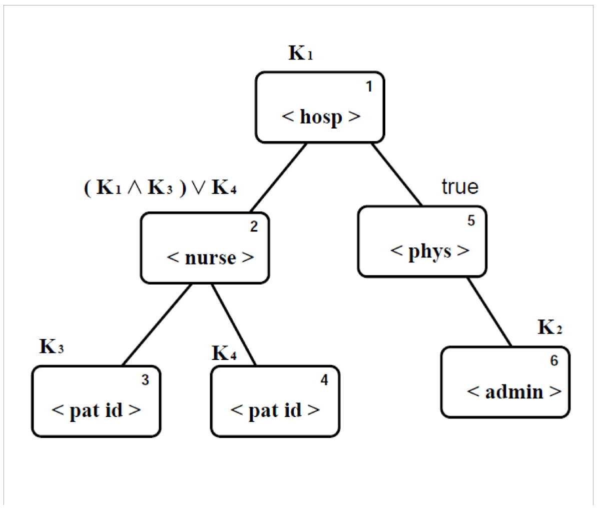


Fig. 2. A tree protection (top) and an equivalent normalized one (bottom).

combination of keys. Normalized protections serve as the basis for producing partially encrypted documents by applying an encryption algorithm repeatedly.

1.6 Conclusion

The formal approach to cryptography often deals with simple, all-or-nothing assertions about security. The computational approach, on the other hand, makes a delicate use of probability and computational complexity. However, one may intuit that the formal assertions are valid in computational models, usually not absolutely but at least with high probability and against adversaries with limited computational power. The author applies it to the study of encryption. He proves that the intuition is correct under substantial but reasonable hypotheses.

2 Introduction to Computational Soundness (cont.)

Speaker: Bogdan Warinschi, University of Bristol

This talk is mainly focused on computational soundness, the bridge between symbolic schemes and computational schemes for security proofs of protocols. The main difference between this talk and the former talk by Martin Abadi is that it is focused on active adversaries while Abadi's talk is focused on passive adversaries. The talk consisted of four main parts: the motivation of computational soundness, computational soundness via blackbox reactive simulation, computational soundness for trace properties, and examples with some additional results.

2.1 Motivation of Computational Soundness

The security proofs of the cryptographic protocols were focused in the symbolic setting since Dolev-Yao model has been proposed in the 80's. But in the symbolic proof, the cryptographers might be unaware of some algebraic properties, and enable the attackers to defeat the protocol. For instance, Warinschi referred to the Needham-Schroeder public key protocol with Lowe's fix (NSL) shown in Fig. 3.

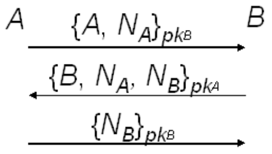


Fig. 3. The Needham-Schroeder Public Key Protocal with Lowe's Fix

This protocol is logically secure. However, in the computational scheme, the adversaries may be able to change the second message from $\{B, N_A, N_B\}$ to $\{C, N_A, N_B\}$ and attack the protocol as in Fig. 4.

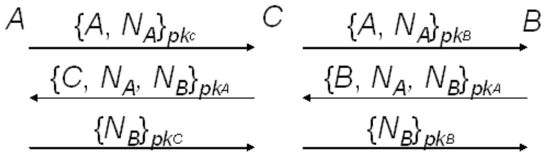


Fig. 4. The Attack of NSL in the Computational Scheme

Computational soundness is the field to bridge this gap by building sufficient security conditions on the primitives used in the implementation such that a protocol which is secure in the symbolic setting is also secure in the computational setting.

2.2 Computational Soundness via Black-Box Reactive Simulation

To bridge the symbolic proof and the computational proof against active adversaries, Warinschi has presented two methods: the simulation approach and the trace mapping approach. In this section, we review the simulation approach, and discuss the trace mapping approach in the next section.

In the simulation approach, we define an ideal cryptoraphic library [3] which offers its users abstract cryptographic operation, such as nonce operation, encryption, decryption, signing, or MACs. This system will handle the entire reactive system, then it contains an abstract network model and covers the case more than Dolev-Yao model does. Some blind spots that were found only in the computational scheme are also included in this system.

In this work, they compare the ideal symbolic cryptographic library with the computational cryptographic library. They can prove that “If cryptographic primitives are secure in the computational cryptographic library, then there exists a simulation such that no probabilistic polynomial time environment can distinguish between the two worlds”.

2.3 Computational Soundness via Trace Mapping

In this method, they model the symbolic state of the group as F_i, and the computation state as G_i. When some participants in the states F_i or G_i send message m_i to other participants, the states change to F_{i+1} or G_{i+1}. We define the series F_i and m_i for the protocol Π and the adversary A as $Tr_s(\Pi, A)$, and define the execution trace $Tr_c(\Pi(R_\Pi), A(R_A))$ for the series G_i and m_i determined by the adversary A and randomness R_Π and R_A.

The main result of this section is that they can prove that

$$\Pr[\exists B, \exists F_c : Tr_c(\Pi(R_\Pi), A(R_A)) = f_c(Tr_s(\Pi, B))] \text{ is overwhelming}$$

which is called “mapping lemma”. We illustrate mapping lemma in Fig. 5.

$$\begin{tikzcd}
F_0 \arrow[rr, "m_0"] \arrow[d, "f_c"] & {} \arrow[d, "f_c"] & F_1 \arrow[rr, "m_1"] \arrow[d, "f_c"] & {} \arrow[d, "f_c"] & F_2 \arrow[rr, "m_2"] \arrow[d, "f_c"] & {} \arrow[d, "f_c"] & F_3 \arrow[rr, "m_3"] \arrow[d, "f_c"] & {} \arrow[d, "f_c"] & \cdots \\
G_0 \arrow[rr, "m_0"] & {} & G_1 \arrow[rr, "m_1"] & {} & G_2 \arrow[rr, "m_2"] & {} & G_3 \arrow[rr, "m_3"] & {} & \cdots
\end{tikzcd}$$

Fig. 5. Mapping Lemma

3 The Game-based Methodology for Computational Security Proofs

Speaker: David Pointcheval, École normale supérieure, LIENS-CNRS-INRIA, France

3.1 Cryptography and Provable Security

Since the beginning of public-key cryptography, with the seminal Diffie-Hellman paper [4], many suitable algorithmic problems for cryptography have been proposed and many cryptographic schemes have been designed, together with more or less heuristic proofs of their security relative to the intractability of the above problems. However, most of those schemes have thereafter been broken. There exist two main frameworks for analyzing the security of cryptographic protocols. The most famous one, among the cryptographic community, is the provable security in the reductionist sense: adversaries are probabilistic polynomial-time Turing machines which try to win a game, specific to the cryptographic primitive or protocol and to the security notion to be satisfied. The computational security is achieved by contradiction: if an adversary can win such an attack game with non-negligible probability, then a well-defined computational assumption is invalid (e.g., one-wayness, intractability of integer factoring, etc.) As a consequence, the actual security relies on the sole validity of the computational assumption. On the other hand, people from formal methods defined formal and abstract models, the so-called Dolev-Yao framework [5], in order to be able to prove the security of cryptographic protocols too.

In complexity theory, such an algorithm which uses the attacker as a subpart in a global algorithm is called a reduction. If this reduction is polynomial, then we can say that the attack of the cryptographic protocol is at least as hard as inverting the function: if one has a polynomial algorithm to solve the latter problem, he can polynomially solve the former one. In the complexity theory framework, a polynomial algorithm is the formalization of efficiency. Therefore, in order to prove the security of a cryptographic protocol, one first needs to make precise the security notion he wants the protocol to achieve: which adversary's goal one wants to be intractable, under which kind of attack.

3.2 Game-based Methodology

The game-playing technique is a general method to structure and unify cryptographic proofs [6]. Its central idea is to view the interaction between an adversary and the cryptosystem as a game, and to study game transformations that preserve security, thus permitting to transform an initial game, that explicitly encodes a security property, into a game where it is easy to bound the advantage of the adversary.

There are some security notions which capture the main practical situations. On the one hand, the goals of the adversary may be various:

– Disclosing the private key of the signer. It is the most serious attack. This attack is termed total break.
– Constructing an efficient algorithm which is able to sign messages with good probability o success. This is called universal forgery.
– Providing a new message-signature pair. This is called existential forgery. The corresponding security level is called existential unforgeability (EUF).
– one-wayness under chosen-plaintext attacks (OW-CPA): where the adversary wants to recover the whole plaintext from just the ciphertext and the public key. This is the weakest scenario.
– semantic security under adaptive chosen-ciphertext attacks (IND-CCA): where the adversary just wants to distinguish which plaintext, between two messages of its choice, has been encrypted, while it can ask any query it wants to

a decryption oracle (except the challenge ciphertext). This is the strongest scenario one can define for encryption (still in our general framework.) Thus, this is our goal when we design a cryptosystem.

Given a family of memories M_i indexed by the natural numbers we say that the difference in the probability of an event A between two games G_1 and G_2 is negligible if and only if:

$$G_1 \approx G_2 := neg(\lambda\eta| \Pr_{G_1,M(\eta)}[A] - \Pr_{G_2,M(\eta)}[A]|) \tag{1}$$

where *neg* is a negligible function and η is our security parameter.

An often used technique to prove that two games are indistinguishable is based on failure events, like the example we introduce in the Game transformations. This technique relies on a fundamental lemma that permits to bound the difference of the probability of a given event in two games by identifying a failure event and arguing that the games behave identically until this event occurs.

One distinction has been widely used for the chosen-ciphertext attacks, for historical reasons: the non-adaptive chosen-ciphertext attacks (CCA1) and adaptive chosen-ciphertext attacks (CCA2). The latter scenario which permits adaptively chosen ciphertexts as queries to the decryption oracle is definitely the strongest attack, and will be named the chosen-ciphertext attack (CCA). One starts with an initial game, chosen to prove some security definition and then reduces this game to a trivial game, to show for example indistinguishability. The reduction is done by showing that some intermediate games are equivalent, from a probabilistic point of view. Reduction proven indistinguishable for an IND-CCA adversary (actually IND-CCA1, and not IND-CCA2) but widely believed for IND-CCA2, without any further analysis of the reduction. The direct reduction methodology: Shoup showed the gap for IND-CCA2, under the one way permutation (OWP) and granted his new game-based methodology. It is proven that the security for IND-CCA2 is guaranteed under the PD-OWP using the game-based methodology.

When we have to do the security proof, we should do as follows:

1. Define goal of adversary
2. Define security model
3. Define complexity assumptions
4. Provide a proof by reduction
5. Check proof
6. Interpret proof

3.3 Identity-based Encryption(IBE)

IBE system was proposed in Boneh et al. [7]. An IBE system can be built from any bilinear map $e : G_1 \times G_1 \longrightarrow G_2$ between two groups G_1, G_2 as long as a variant of the Computational Diffie-Hellman problem if G_1 is hard. We use the Weil pairing on elliptic curves as an example of such a map.

We say that an identity-based encryption scheme $\mathcal{E}$ is semantically secure against an adaptive chosen ciphertext attack (IND-ID-CCA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following IND-ID-CCA game. Note that the standard definition of chosen ciphertext security (IND-CCA) is the same as above except that there are no private key extraction queries and the adversary is challenged on a random public key (rather than a public key of her choice). Private key extraction queries are related to the definition of chosen ciphertext security in the multiuser setting. After all, the definition involves multiple public keys belonging to multiple users. This does not hold in the identity-based settings, IND-ID-CCA, since the adversary gets to choose which public keys to corrupt during the attack.

Here is a description of the ID-based encryption scheme

Setup: The authority generates a master secret key *msk*, and publishes the public parameters, *PK*

Extraction: Given an identity *ID*, the authority computes the private key *sk* granted the master secret key *msk*

Encryption: Anyone can encrypt a message m for a user *ID* using only m, *ID* and the public parameters *PK*

Decryption: Given a ciphertext, only the user *ID* can recover the plaintext, using *sk*

3.4 Security Analysis

Definition of IND-ID-CCA Security:

- A receives the global parameters
- A asks any extraction-query, and any decryption-query
- A outputs a target identity ID' and two messages (m_0, m_1)
- The challenger flips a bit b, and encrypts m_b for ID', obtaining c'
- A asks any extraction-query, and any decryption-query
- A outputs its guess b' for b

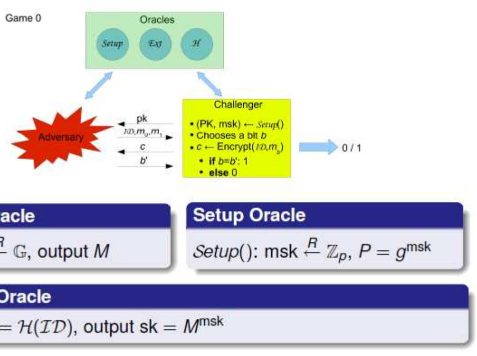
Restriction: ID' is never asked to the extraction oracle, and (ID', c') are never asked to the decryption oracle.

It can be shown that there is a reduction between breaking the BF-IBE in the Semantic Security model and the BDHP problem.

Boneh et. al's IBE security analysis is as follows:

Theorem 1. *The BF-IBE is IND-ID-CPA secure under the DBDH problem, in the random oracle model by masking m with* $H(K) : B = m \oplus H(K)$*, the BF-IBE is IND-ID-CPA secure under the CBDH problem, in the random oracle model.*

Theorem 2. *The BLS signature achieves EUF-CMA security, under the CDH assumption in G, in the Random Oracle Model.*



Game 0
Oracles
$\mathcal{S}etup$
$\mathcal{E}xt$
$\mathcal{H}$
Adversary
pk
$\mathcal{ID}, m_0, m_1$
c
b'
Challenger
• (PK, msk) ← $\mathcal{S}etup$()
• Chooses a bit b
• c ← Encrypt($\mathcal{ID}, m_b$)
• **if** b=b': 1
• **else** 0
0 / 1
Random Oracle
$\mathcal{H}(\mathcal{ID})$: $M \overset{R}{\leftarrow} \mathbb{G}$, output M
Setup Oracle
$\mathcal{S}etup()$: msk $\overset{R}{\leftarrow} \mathbb{Z}_p$, $P = g^{\mathsf{msk}}$
Extraction Oracle
$\mathcal{E}xt(\mathcal{ID})$: $M = \mathcal{H}(\mathcal{ID})$, output sk $= M^{\mathsf{msk}}$

Fig. 6. The real attack

3.5 Conclusion

The ID-based encryption system has chosen ciphertext security in the random oracle model assuming BDH, a natural analogous of the computational Diffie-Hellman problem. One might hope to use the techniques of Cramer-Shoup to provide chosen ciphertext security based on DDH. In this talk, the speaker showed how to use the game-based methodology which uses a sequence of games to do the security proof. The transition hops are simple and easy to check. It leads to easy-to-read and easy-to-verify security proofs: Some mistakes have been found granted this methodology with the analysis of OAEP. Some security analyses became possible to handle using the analysis of EKE. This approach can be automated by CryptoVerif.

4 On Reactive Simulatability

Speaker: Matthias Berg, Universität des Saarlandes

The reactive simulatability model as introduced by Backes, Pfitzmann and Waidner in [8] is a framework for security proofs and provides guarantees for the composability of cryptographic protocols. Section 4.1 motivates the intoduction of the reactive simulatability model, which is then presented in section 4.2. Composition in the model is considered in section 4.3, and the abstract model is viewed in more detail in section 4.4. Section 4.5 summarizes the talk's main points.

4.1 Motivation

Cryptographic protocols make use of primitives such as encryption or digital signatures. Instead of trying to prove security for a whole protocol, it could be easier to prove security for the primitives first, and use abstract primitives that give abstract security guarantees in the security proof of the protocol. The proofs of the abstract primitives and the abstract protocols composed of them should then carry over to concrete primitives and protocols used in the real world.

For a sound abstraction, several tasks have to be accomplished.

– The real world has to be mapped onto a precise system model that includes the network, the adversary, scheduling, and concurrency with other protocols.
– As the model includes an abstract world and a realistic world, methods for reasoning about both real-world cryptography and its abstract, formal representation have to be provided.
– A definition should be given that details what abstractions are "good" in a sense that they are intuitive but still allow the use of convenient proof techniques.
– Protocols are usually composed of and run together with other protocols, so guarantees should hold also under composition.
– The abstraction should preservation arbitrary security properties.

4.2 Overview of the Framework

The reactive simulatability framework consists of a precise system model that allows cryptographic and abstract operations. The real system consists of a number of honest parties H making use of cryptographic protocols M_i, and an adversary $\mathcal{A}$. The ideal system consists of a number of honest parties H using abstract protocols $\mathcal{F}$, an adversary $\mathcal{A}'$, and a simulator $\mathcal{S}$ that makes the use of abstract protocols transparent to the adversary. To each cryptographic protocol in the real system there is an abstract protocol in the ideal system which should provide the same functionality. This duality requires sound, Dolev-Yao style symbolic abstractions and makes sound security proofs for cryptographic protocols possible.

The idea behind the two worlds is to define security relative to an ideal task, so that everything that can happen in the real system can also happen to the ideal system, which is secure by design. Then, the real system is said to be *as secure as* the ideal system.

It accommodates several versions of simulatability, and provides appropriate composition theorems. For standard simulatability, it is required that

$$\forall \mathcal{A} \forall H \exists \mathcal{A}' \; view_{real}(H) \approx view_{ideal}(H),$$

where $\approx$ is the indistinguishability of random variables.

Two families of random variables $(v), (v')$ are said to be indistinguishable if for all PPT distinguishers D

$$|\Pr[D(1^k, v_k) = 1] - \Pr[D(1^k, v'_k) = 1$$

is negligible in k.

The stronger notion of universal simulatability $\mathcal{A}'$ cannot depend on the honest users

$$\forall \mathcal{A} \forall H \exists \mathcal{A}' \, view_{real}(H) \approx view_{ideal}(H),$$

while in black-box simulatability $\mathcal{A}'$ consists of a simulator $\mathcal{S}$ that uses $\mathcal{A}$ so that the protocol is secure if

$$\exists \mathcal{S} \forall H \forall A \, view_{real}(H) \approx view_{ideal}(H).$$

4.3 Composition

To guarantee composition of protocols, first the combination of the machines which represent the protocols has to be defined. This combination is associative and retains the polynomial running time of its component sub-machines as well as their views. In this way, even hybrid machines of abstract and real machines can be constructed.

The relationship *as secure as* ($\geq$) is then transitive. The composition theorem states that if a real protocol is *as secure as* an ideal protocol, then any instance of the real protocol used by any machine can be replaced by an instance of the abstract protocol.

A proof of this theorem can be given via hybrid arguments, where machines are unified in changing configurations. An abstraction of this proof is shown in figure 7.

4.4 The Abstract System

The abstract system of the BPW model uses Dolev-Yao-style term algebras, for which well-developed proof theories exist.

The main challenge of this approach is to invent a standard interface that represents the real library, which uses bit strings, and the ideal library, which uses abstract data types, in the same way to higher protocols. Furthermore, because this behavior cannot be modeled in the abstract world, users must be prevented from abusing cryptographic objects, and the decision must be made which imperfections between the two worlds are allowed.

Characteristic of the BPW model is the library of standard crypto primitives implemented both in the real and the abstract world, which implements the required unified interface. The honest users and the adversary manipulate the messages indirectly using *handles*. The model tracks the sending of messages and the knowledge of the participants about the information represented by the handles.

To accomplish this, it provides cryptographic functions for message construction, as well as functions to access and transmit messages. Messages from parties are transmitted via the functions to the adversary, who can choose to deliver them further. The interface for the adversary provides him with additional capabilities, for example to create invalid messages or ciphertexts.

The BPW model has several differences to the standard Dolev-Yao models. To incorporate the necessary imperfections cryptographic protocols have in the

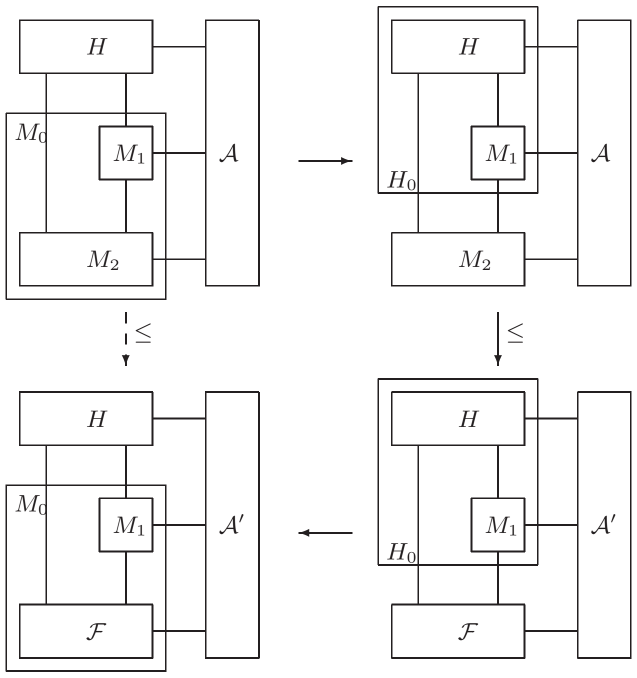


Fig. 7. Proof of the simple composition theorem

real world, each encryption is tagged with the length of the encrypted plaintext. Furthermore, it provides for the encryption of incorrect messages by $\mathcal{A}$, stateful signatures and a slightly restricted usage of keys for symmetric encryption.

4.5 Summary

The reactive simulatability model links cryptography to the use of formal methods by extending the classic Dolev-Yao model and correlating it with a model representing the real world. The structure of the model also makes it possible to give guarantees about the composability of protocols.

5 Introduction to Universally Composable Security

Speaker: Olivier Pereira, Université catholique de Louvain

Protocols are usually executed together with other protocols, so it would be helpful if a stand-alone proof was preserved under composition. How a model describing general protocols can be constructed is shown in section 5.1 by using secure function evaluation as an example. A simple model providing universal composability is described in section 5.2 and afterwards generalized in section 5.3. The universal composition theorem is stated in section 5.4. Section 5.5 summarizes the talk's main points.

5.1 Motivation

Since we want to derive a security notion for general cryptographic protocols, we consider the most general protocol problem: secure function evaluation (SFE). SFE consists of parties P_i, each with a private input x_i, that want to compute the output y_i of a function f on their inputs. The protocol should give only the output to each party, and nothing else.

This general description can be used to derive all protocols that do not rely on keeping an internal state between multiple activations. Authentic communication from a party P_1 to a party P_2 in the presence of an adversary $\mathcal{A} = P_3$ can be modeled as $(-, m, m) = f(m, -, -)$. Secure communication could be $(-, m, |m|) = f(m, -, -)$, while a key exchange would be $(k, k, |k|) = f(-, -, -)$.

We first take a look at the most simple scenario, SFE between two parties (one of which may be malicious), using authenticated channels.

A protocol should fulfill at least two requirements: *Correctness*, so each party P_i receives y_i, and *privacy*, so no party learns anything about the other parties' inputs.

However, for the function XOR $(x_1 \oplus x_2, x_1 \oplus x_2) = f(x_1, x_2)$, correctness prevents privacy. This is necessary, but makes the property of *input independence* desirable, so that no party cannot choose its input as a function of the input of another party (which would enable it to dictate the output value).

This example shows that for a general model it is difficult to list all properties a protocol should have. Simulation-based security follows a different approach, which is described below.

5.2 The Model

To get a generic security definition that provides a unified framework for all protocol tasks, we define an ideal world, in which a trusted component evaluates the function securely. A protocol is then secure if it emulates this behavior. This seems to be a natural way to define security, and captures all the requirements derived above for SFE.

The ideal world consists of honest parties P_i, the secure functionality $\mathcal{F}$, and the adversary $\mathcal{A}$. By definition, everything the adversary does in this world is harmless.

The real world consists of honest parties P_i and the adversary $\mathcal{A}$. A real-world protocol Π is secure if it *emulates* the behavior of $\mathcal{F}$ in the ideal world, that is if $\forall \mathcal{A}$ in the real world $\exists \mathcal{A}'$ in the ideal world, such that the behaviors of the two systems cannot be distinguished. This distinguisher is the environment $\mathcal{E}$, which knows the inputs of all parties.

In a real-world protocol execution:

1. $\mathcal{E}$ chooses the input for P_i and $\mathcal{A}$
2. P_i and $\mathcal{A}$ run the protocol
3. P_i and $\mathcal{A}$ send their output to $\mathcal{E}$
4. $\mathcal{E}$ outputs a bit

In the ideal world:

1. $\mathcal{E}$ chooses the input for P_i and $\mathcal{A}'$
2. P_i forwards its input to $\mathcal{F}$
3. $\mathcal{A}$ communicates with $\mathcal{F}$
4. $\mathcal{F}$ sends its output to $\mathcal{A}'$
5. When $\mathcal{A}'$ says "OK", $\mathcal{F}$ sends the output to P_i
6. $\mathcal{A}'$ and P_i send their output to $\mathcal{E}$
7. $\mathcal{E}$ outputs a bit

If $\mathcal{F}$ would send its output to P_i without waiting for the OK by $\mathcal{A}'$, P_i will always receive the output, while typically in the real world the adversary is able to make a protocol run fail.

A protocol Π is secure with respect to $\mathcal{F}$ if $\forall \mathcal{A}$ in the real world $\exists \mathcal{A}'$ in the ideal world such that $\forall \mathcal{E}$ $\textsc{Exec}(P_i, \mathcal{A}, \mathcal{E}) \approx \textsc{Exec}(\mathcal{F}, \mathcal{A}', \mathcal{E})$ where $\approx$ can be chosen in a number of ways, yielding different security guarantees. Since $\mathcal{A}'$ simulates the real-world execution with $\mathcal{A}$, it is called a *simulator*

5.3 Generalization

Now that a reasonable security definition has been found, we want to have more general protocol tasks. For this, we can do several things:

– allow more than two parties
– route all communications from the parties through the functionalities to the adversary; in this way an unreliable network can be simulated
– allow $\mathcal{F}$ to be any process instead of only a SFE, so $\mathcal{F}$ can leak information to the adversary, keep state, etc.
– allow $\mathcal{A}$ to interact freely with $\mathcal{E}$ to obtain concurrent executions

5.4 Composition

We want protocols to retain their security when executed together with other protocols. Different composition modes can be considered: We can consider composition of a protocol with itself or with arbitrary other protocols. Timing can be sequential, non-concurrent, parallel, or concurrent, and the number of executions can be constant, polynomial or unbounded. Instances can share states, and the inputs can be chosen by the adversary or the environment. We want protocols to preserve their ideal-world behavior under composition, even with protocols that interact with them in an arbitrary manner.

Universal composition can be used to cover all the abovementioned cases, essentially treating a protocol as a procedure for other protocols. To accomplish this, define for a protocol ρ using an ideal functionality ϕ, and a protocol π that provides the same interface as ϕ the protocol $\rho^{\pi/\phi}$ where all instances of ϕ are replaced with instances of π.

The *universal composition theorem* then states that ideal functionalities can be replaced with secure protocols: If π emulates ϕ, then $\rho^{\pi/\phi}$ emulates ρ.

5.5 Conclusion

The real-world / ideal-world paradigm allows us to separate security from the communications model and develop composition theorems. The definition given in this talk however is still not satisfying, as all instances of the protocol have their own state, which does not allow for notions such as long-term secrets shared between instances. Therefore, we need composition with joint state.

6 Simulation-based Security and Joint State Theorems in the IITM model

Speaker: Ralf Küsters, Universität Trier

The simulation-based inexhaustible interactive Turing-machine (IITM) model deals with the problem that a bounded running time poses in the UC model and provides for easier composability with joint states. Section 6.1 describes three problematic aspects of simulation-based models for which the IITM model, presented in section 6.2, tries to find good solutions. Section 6.3 covers systems with joint states, for which the IITM model has an especially simple way of formulating composition theorems. Setion 6.4 gives a short repetition of the talk's main points.

6.1 Subtleties in Simulation-based Models

Simulation-based models differ in several aspects. We now take a closer look at three of them, the definition of simulatability, the master process, and the runtime of the ITMs.

Definitions of simulatability There are several different simulation-based models, with different flavors of composability. In each model, an ideal world *ideal* is defined, in which an ideal adversary I interacts with an ideal functionality $\mathcal{F}$. By construction, the adversary can do nothing that would be considered harmful. A protocol Π in the real world *real* is considered secure, if every interaction of a real-word adversary $\mathcal{A}$ with the protocol is indistinguishable from that of an ideal adversary I with the ideal functionality, and is thus harmless. The distinguisher is an environment $\mathcal{E}$, which appears in both worlds.

In the UC model, Π UC-realizes $\mathcal{F}$ if $\forall \mathcal{A} \exists I \forall \mathcal{E}\ ideal \equiv real$.

For strong black-box simulatability, the ideal adversary I consists of the real-world adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ that "translates" between him and the ideal world. Π SBB-realizes $\mathcal{F}$ if $\exists \mathcal{S} \forall \mathcal{A} \forall \mathcal{E}\ ideal \equiv real$.

Strong simulatability requires that $\exists \mathcal{S} \forall \mathcal{E}\ ideal \equiv real$, where the environment contains the adversary as a sub-machine.

The master process The master process is triggered if no other process can go. The question who takes the role of the master process can be answered differently to obtain different notions of security. In particular, the notions of SS, SBB, UC, and WBB (weak black-box simulatability) are equivalent if the environment takes over the master role, and the *forwarder* property holds. The forwarder propery is fulfilled if a process forwarding messages can be placed between any two processes without changing the behavior of the system, which may not be possible if the running time of the forwarder must be chosed before that of the receiver or the sender [9]. This is the case in the IITM model, which suggests that a good definition of simulation-based security has been found.

Runtime of the ITMs In the UC model, the total runtime of components is polynomially bounded in the security parameter alone and independent of external input. This leads to exhaustible ITMs, which means that an ITM with can send messages to another ITM with a smaller runtime until the second ITM has "used up" its runtime and does not respond to further messages. This bounded runtime also imposes an artificial bound on the length and number of messages that can be handled by protocols.

The fast that the runtime of an ITM is bounded also implies that the parallel composition of two or more protocols cannot be simulated by one ITM, because the adversary can exhaust one machine and still communicate with the other, something that is not possible for the combined machine.

6.2 The IITM Model

The IITM model consists of the general computational model, which defines IITMs and their interaction in systems of IITMs, and the simulation-based security model, which defines security notions and gives composition theorems.

The computational model An inexhaustible interactive Turing-machine (IITM) is an ITM which runtime may depend on the length of the input. It can be activated an unbounded number of times, and can perform PPT computations in every activation. This allows the composition of IITMs, something which would not be possible if the machines could be forced to stop. However, combining ITMs is not easy if their runtime depends on the length of input messages, because non-terminating systems are possible. Imposing a global polynomial bound is impossible, as it enables $\mathcal{E}$ to distinguish between real and ideal.

This probem is solved by dividing the imput tapes of IITMs into two kinds: consuming and enriching ones. In each activation, the IITM may perform a computation polynomially bounded in the length of the current input plus the length of current configuration plus the security parameter. This enables the ITM to read every message and scan the entire configuration in every activation, and prevents exhaustion.

The length of output and configuration is polynomially bounded in the security parameter plus the length of the input received on the enriching tapes so far.

IITMs also use a generic addressing mechanism, which means that no specific mechanism is fixed. This is useful for defining systems of IITMs. When multiple copies of a machine M exist, $\underline{M}$ denotes the union of these machines, so that they can be addressed by the tuple $(\underline{M}, id)$, where id is used to address a specific instance within $\underline{M}$.

A system of IITMs S consists of a number of IITMs, together with a number of IITMs that can dynamically generate an unbounded number of copies of themselves, denoted by "!": $S = M_1 \| \cdots \| M_n \| !M'_1 \| \cdots \| !M'_m$.

A system is *well-formed* if the graph of IITMs induced by the enriching tapes is acyclic. This allows us to prove that a well-formed system is a sensible model of computation.

Lemma 1. *Well-formed systems run in PPT.*

We can also prove the existence of a forwarder IITM, so that the notions of security mentioned in section 6.1 collapse.

Lemma 2. *There exists a* forwarder *IITM D such that $P \| Q = P \| D \| Q$, where D is independent of P and Q, and all tapes are enriching.*

The following lemma is needed for the joint state theorem.

Lemma 3. *Given systems Q_1, Q_2 with $Q_1 \| Q_2$ well-formed, then there exists an IITM M s.t. $Q_1 \| Q_2 = Q_1 \| M$*

We now describe the generic addressing mechanism. IITMs run in one of two modes, *check address* or *compute*.

If a message is addressed to a machine M_1, all existing instances check (in order of creation) if they accept the message. If no existing copy accepts, a new copy is created. If it also does not accept, the master is activated.

Simulation-based security Security is defined similarly to UC and black-box simulatability.

Definition 1 (Strong Simulatability). *A protocol Π securely realizes an ideal functionality $\mathcal{F}$ ($\Pi \leq \mathcal{F}$) if and only if $\exists \mathcal{S} \forall \mathcal{E} : \mathcal{E} \| \Pi \equiv \mathcal{E} \| \mathcal{S} \| \mathcal{F}$.*

The composition theorem in the IITM model is formulated as follows.

Theorem 3. $P_1 \leq \mathcal{F}_1 \wedge P_2 \leq \mathcal{F}_2 \Rightarrow P_1 \| P_2 \leq F_1 \| F_2$
$P_1 \leq \mathcal{F}_1 \Rightarrow !P_1 \leq !F_1$

It should be noted that in UC, $\mathcal{E}$ cannot connect to $\mathcal{F}_2$.

6.3 Joint State

Suppose a protocol Π uses $\mathcal{F}_{PKE}$ to securely realize $\mathcal{F}_{KE}$. If several instances of Π run concurrently, they all use their own version of $\mathcal{F}_{PKE}$, which means that they all have their own key pair. This can be avoided by letting $\mathcal{F}_{PKE}$ be shared between the instances of Π, which is made possible by the following theorem.

Theorem 4 (General Joint State Theorem). *If a protocol $\hat{P} = P^{JS} \| \mathcal{F}$ securely implements a shared functionality $\underline{\mathcal{F}}$, then $\underline{\mathcal{F}}$ can be replaced by $\hat{P}$ in any protocol Q using $\underline{\mathcal{F}}$.*
$\hat{P} \leq !\underline{\mathcal{F}} \Rightarrow Q \| \hat{P} \leq Q \| !\underline{\mathcal{F}}$

Example 1 (Iterative application of the general joint state theorem). Consider the following scenario, where a functionality Q uses two instances of a functionality $\underline{Q'}$, each of which uses two instances of a PKE functionality $\underline{\underline{\mathcal{F}}}$. The application of the theorem is depicted in figure 8.

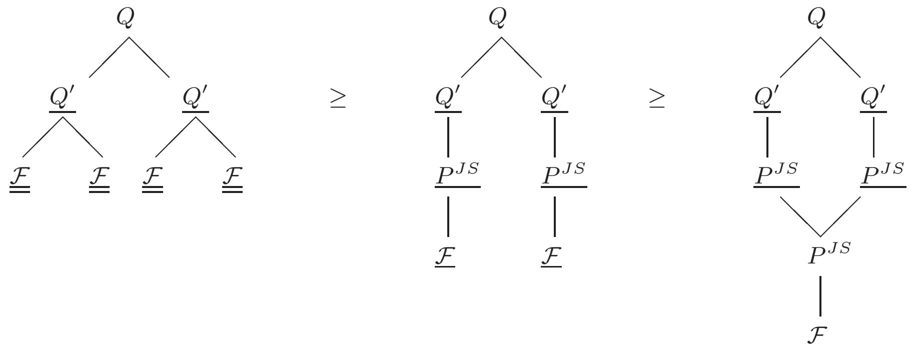


Fig. 8. Example of GJS theorem application

6.4 Conclusion

Simulation-based security is useful due to its modular design and simpler analysis. Models for simulation-based security do not have to be complicated, as shown by the IITM model, which gives a simple way to deal with composability even when using joint states.

7 Cryptographic Applications of Indifferentiability via Leaking Random Oracle Models

Speaker: Kazuo Ohta, University of Electro-Communications

The indifferentiability is useful for the random oracle methodology and the design and security analysis of hash functions. However, the Merkle-Damgård hashing is indifferent from the random oracle. Thus, there exists some protocols that may be insecure if the random oracle is instantiated by the Merkle-Damgård hashing. So, this talk proposes three approaches to rescue the Merkle-Damgård hashing and concentrate on the second approach. The definition of the indifferentiabilitySection is shown in section 7.1. Section 7.2 overviews the negative result about the Merkle-Damgård hashing. Section 7.3 shows the three approaches to rescue the Merkle-Damgård hashing and describes more detail of the second approach. Section 7.4 summarizes the talk's main points.

7.1 Indifferentiability

The indifferentiability framework for general case was introduced by Maurer et al. in 2004, and for hashing functions was proposed by Coron et al. in2005. This concept is a generalization of the indistinguishability. If a primitive $\mathcal{U}$ is indifferentiable from other primitive $\mathcal{V}$, denoted $\mathcal{U} \sqsubset \mathcal{V}$, and a cryptosystem $C(\mathcal{U})$ is secure, then a cryptosystem $C(\mathcal{V})$ is also secure, denoted $C(\mathcal{U}) > C(\mathcal{V})$. And a definition of the indifferentiability for hash functions is showed in Fig:9. A hashing function $\mathcal{H}$ is indifferentiable from an ideal primitive $\mathcal{F}$ ($\mathcal{H} \sqsubset \mathcal{F}$), if for any distinguisher $\mathcal{D}$ with an output 0 or 1, there is a simulator $\mathcal{S}$ such that the advantage $|Pr[\mathcal{D}(\mathcal{H}, \mathcal{G}) = 1] - Pr[\mathcal{D}(\mathcal{F}, \mathcal{S}) = 1]|$ is negligible.

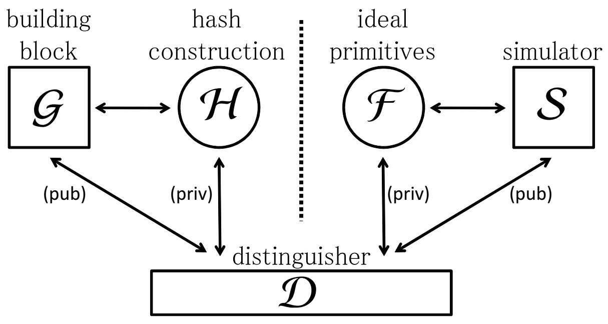


Fig. 9. Indifferentiability for hash

7.2 Negative Result about the Merkle-Damgård hashing

Fig:10 is the original Merkle-Damgård construction denoted by MD^h, where h is a compression function. An input message M is divided into small message blocks m_i. This mode of operation is adopted by widely used hash functions such as MD5, SHA-1 and SHA-256. Coron et al. proved that the Merkle-Damgård (MD) hashing is not indifferentiable from the random oracle ($\mathcal{RO}$), that is $MD^{\mathrm{FIL}\mathcal{RO}} \not\sqsubset \mathcal{RO}$, where FIL$\mathcal{RO}$ is fixed input-length $\mathcal{RO}$. This means that there exists a protocol secure in the random oracle but insecure if $\mathcal{RO}$ is instantiated by the MD hashing. This fact is due to an extension attack shown in Fig:11. This extension attack constructs a distinguisher $\mathcal{D}$ between the MD hashing and $\mathcal{RO}$. $\mathcal{D}$ sends a message M_1 to a mode of operation component or message extension

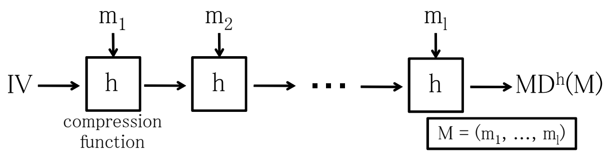


Fig. 10. Merkle-Damgård hashing

first. In the real world, *MD* sends IV and M_1 to a compression function, and obtains a value y_1. *MD* returns y_1 to $\mathcal{D}$. In the ideal world, $\mathcal{RO}$ returns y_1 as an answer to M_1. In both cases, since y_1 is an output from $\mathcal{RO}$, there is no difference for $\mathcal{D}$ up to this point. $\mathcal{D}$ sends y_1 and a message m to a compression function on a public channel, and obtains y_2. $\mathcal{D}$ constructs a new message M_2 by concatenating M_1 with m, and sends M_2 to a mode of operation component, *MD* exists in the left side, the real world, and $\mathcal{RO}$ exists in the right side, the ideal world. In the real world, *MD* sends IV and M_2 following the construction procedure of MD^h. After the compression function returns y_3, *MD* transfers it to $\mathcal{D}$. In the ideal world, $\mathcal{RO}$ returns y_3 as an answer to M_2. In the left side $y_2 = y_3$ holds exactly, while in the right side y_3 is independently chosen from y_2. Thus $\mathcal{D}$ can decide that if the equality holds, output 1 and it does not hold, output 0. This is a distinguishing strategy that can be used by $\mathcal{D}$.

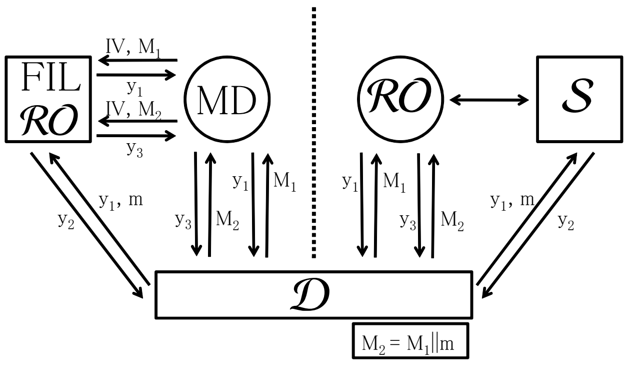


Fig. 11. Extension Attack

7.3 Approach

There are three approaches to rescue the *MD* hashing. The first is to use modified the *MD* hashing. However, this approach cannot rescue the original *MD* hashing. The second is to use leaking Random Oracle models. The third is to use indifferentiability with conditions. Afterward, more detail of the second approach is described.

Main strategy of this approach is two steps. The first step is to find an ideal primitive $\widetilde{\mathcal{RO}}$ from which $MD^{\mathrm{FIL}\mathcal{RO}}$ is indifferentiable. The second step is to prove that a cryptosystem $\mathcal{C}$ is secure in the $\widetilde{\mathcal{RO}}$ model.

A example of $\widetilde{\mathcal{RO}}$ is the leaky random oracle $\mathcal{LRO}$ model. This is a weakened $\mathcal{RO}$ model that was created to analyze security against leakage of the hash list. The majority of signature schemes and Cramer-Shoup encryption are secure within this LRO model. But OAEP and Kurosawa and Desmedt-PKE are insecure. $\mathcal{LRO}$ consists of the random oracle $\mathcal{RO}$ and the leak oracle $\mathcal{LO}$. There are two kinds of queries, the hash query and the leak query. The hash query is issued to $\mathcal{RO}$. $\mathcal{RO}$ responds with hash value y_1 to query x_1. Here the pair of $(x_1,\, y_1)$ is memorized in the table of $\mathcal{RO}$. The leak query is issued to the $\mathcal{LO}$. The leak signal is a trigger for $\mathcal{LO}$ which reveals the entire information of the table. For this construction, $MD^{\mathrm{FIL}\mathcal{RO}}$ is indifferentiable from $\mathcal{LRO}$. And, FDH is still secure in the $\mathcal{LRO}$ model. However, OAEP is not one-way in the $\mathcal{LRO}$ model.

The second example is the traceable random oracle $\mathcal{TRO}$ model. TRO consists of $\mathcal{RO}$ and the trace oracle $\mathcal{TO}$. There are two kinds of queries, the hash query and the trace query. $\mathcal{RO}$ have the same behavior as one in $\mathcal{LRO}$. The trace query is issued to $\mathcal{TO}$. When the trace query is y_2, $\mathcal{TO}$ reveals the input x_2 corresponding to y_2 by accessing the table. When the trace query is y_1, $\mathcal{TO}$ reveals all inputs x_1 and x_2 that correspond to y_1. When the trace query is y' and there is no input corresponding to y' in the table, this symbol is returned. Then, $MD^{\mathrm{FIL}\mathcal{RO}}$ is indifferentiable from $\mathcal{TRO}$. And, OAEP is secure in the $\mathcal{TRO}$ model. However, RSA-KEM is not IND-CPA in the $\mathcal{TRO}$ model.

The final example is the extension attack simulatable random oracle $\mathcal{ERO}$ model. $\mathcal{ERO}$ consists of $\mathcal{RO}$ and extension attack oracle $\mathcal{EO}$. There are two kinds of queries, the hash query and the extension attack query. $\mathcal{RO}$ have the same behavior as one in $\mathcal{LRO}$ and $\mathcal{TRO}$. The extension attack query is issued to $\mathcal{EO}$. When the query is x' and y_1, $\mathcal{EO}$ concatenates query x_1 with x', sends the result to RO, and obtains y_2 from $\mathcal{RO}$. RO memorizes x_1 concatenated with x' and y_2 on $\mathcal{RO}$'s table. EO memorizes (x', y_1, y_2) on the table, and returns y_2 as an answer to the query. When the extension attack query is x'' and y', $\mathcal{EO}$ returns y'' without calling $\mathcal{RO}$ since y' is new. Then, $MD^{\mathrm{FIL}\mathcal{RO}}$ is indifferentiable from $\mathcal{ERO}$. And, RSA-KEM is secure in the $\mathcal{ERO}$ model.

Relation around $\mathcal{RO}$, $MD^{\mathrm{FIL}\mathcal{RO}}$ and these weakened $\mathcal{RO}$s is as follow.

$$\mathcal{RO} \sqsubset MD^{\mathrm{FIL}\mathcal{RO}} \sqsubset \mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}.$$

And,

$$\mathcal{RO} \not\sqsupset MD^{\mathrm{FIL}\mathcal{RO}}, \mathcal{ERO} \not\sqsupset \mathcal{TRO} \not\sqsupset \mathcal{LRO}.$$

7.4 Summary

The indifferentiability is a useful concept for discussing the security of composed cryptosystems as well as the UC framework. This theory gives a negative result on the original *MD* construction. However, practical protocols are provably secure even with the original *MD*. An approach to rescue *MD* is to prove that by considering various leaking $\mathcal{RO}$ models. More preciously, we prove that the original *MD* hashing is indifferentiable from the leaking $\mathcal{RO}$, and the protocol is secure within the leaking $\mathcal{RO}$. The theory of indifferentiability ensures the security of these protocols under the assumption of the FIL $\mathcal{RO}$ compression function.

8 CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols

Speaker: Bruno Blanchet, University of Electro-Communications

Proving the security of cryptographic protocols are difficult and error-prone. So several provers for cryptographic protocols were provided. This talk introduces a computationally sound mechanized prover, CryptoVerif. Section 8.1 shows approaches to prove the security of cryptographic protocols. Section 8.2 overviews CryptoVerif. Experiment results of CryptoVerif are shown in Section 8.3. Section 8.4 summarizes the talk's main points.

8.1 Automatic Proof of Security

There are two security model for security protocols. One is the computational model and the other is the formal model, called the Dolev-Yao model. In general, proofs in the computational model are done manually. On the other hand, proofs in the formal model can be done automatically. It is an important objective to achieve the automatic provability under the realistic computational assumptions.

There are two approaches for the automatic proof of cryptographic protocols in a computational model. One is the indirect approach. This approach is to make a Dolev-Yao proof and use a theorem that shows the soundness of the Dolev-Yao approach with respect to the computational model. The pioneers of this approach are Abadi and Rogaway, and many researchers pursued. The other is the direct approach, that is, we design automatic tools for proving protocols in a computational model. This approach pioneered by Laud. This talk is focused on the direct approach.

8.2 Overview of CryptoVerif

The basic idea of CyrptoVerif is same as in Shoup's method and Bellare and Rogaway's method. The proof is sequence of games as follows.

– The first game is the real protocol.

– One goes from one game to the next by syntactic transformations or by applying the definition of security of a cryptographic primitive.
– The last game is ideal, that is, the security property is obvious from the form of the game.

Games are formalized in a process calculus adapted from the pi calculus showed by Fig:12. The semantics is purely probabilistic. All processes run in polynomial time.

$$
\begin{array}{lll}
M, N & ::= & \qquad\qquad\qquad\qquad\qquad\qquad \textit{terms} \\
 & i & \quad \textit{replication index} \\
 & x[M_1, ..., M_m] & \quad \textit{variable access} \\
 & f(M_1, ..., M_m) & \quad \textit{function application} \\
Q & ::= & \qquad\qquad\qquad\qquad\qquad\qquad \textit{input process} \\
 & 0 & \quad \textit{nil} \\
 & Q|Q' & \quad \textit{parallel composition} \\
 & !^{i \leq n} Q & \quad \textit{replication } n \textit{ times} \\
 & \mathsf{newChannel}\ c; Q & \quad \textit{channel restriction} \\
 & \overline{c[M_1, \ldots, M_l]}(x_1[\tilde{i}] : T_1, \ldots, x_k[\tilde{i}] : T_k); P & \quad \textit{input} \\
P & ::= & \qquad\qquad\qquad\qquad\qquad\qquad \textit{output process} \\
 & c[M_1, \ldots, Ml]\langle N_1, \ldots, N_k \rangle; Q & \quad \textit{output} \\
 & \mathsf{new}\ x[i_1, \ldots, i_m] : T; P & \quad \textit{random number} \\
 & \mathsf{let}\ x[i_1, \ldots, i_m] : T = M\ \mathsf{in}\ P & \quad \textit{assignment} \\
 & \mathsf{if\ defined}\,(M_1, \ldots, M_l) \wedge M\ \mathsf{then}\ P\ \mathsf{else}\ P_0 & \quad \textit{conditional} \\
 & \mathsf{find}\,(\oplus_{j=1}^{m} u_{j1}[\tilde{i}] \leq nj1, ..., u_{jm_j}[\tilde{i}] \leq n_{jm_j} & \\
 & \quad \mathsf{suchthat\ defined}(M_{j1}, ..., M_{jl_j}) \wedge M_j\ \textit{then}\ P_j) & \\
 & \qquad \mathsf{else}\ P & \quad \textit{array lookup}
\end{array}
$$

Fig. 12. Syntax of the process calculus (game)

Game translations are based on observational equivalence between processes. Two processes Q_0, Q_1 are observational equivalent, denoted $Q_0 \approx Q_1$ when the adversary has a negligible probability of distinguishing them. In the formal definition, the adversary is represented by an acceptable evaluation context $C ::= [\,]\ C|Q\ Q|C\ \mathsf{newChannel}\ c; C$. Observational equivalence is an equivalence relation and contextual, that is $Q_0 \approx Q_1$ implies $C[Q_0] \approx C[Q_1]$ where C is any acceptable evaluation context.

The basic proof technique is to transform a game G_0 into an observational equivalent game using following properties.

– Observational equivalences $L \approx R$ given as axioms and that come from security assumptions on primitives. These equivalences are used inside a context:

$$G_1 \approx C[L] \approx C[R] \approx G_2.$$

– Syntactic transformations such as simplification, expansion of assignments.

We obtain a sequence of games $G_0 \approx G_1 \approx \ldots \approx G_m$ which implies $G_0 \approx G_m$. If some equivalence or trace property holds with overwheliming probability in G_m, then it also holds with overwhelming probability in G_0. The followings are more detail of game translations.

– Syntactic transformations
  - Single assignment renaming: when a variable is assigned at several places, rename it with a distinct name for each assignment. This translation is not completely trivial because of array references.
  - Expansion of assignments: replacing a variable with its value. This translation is not also completely trivial because of array references.
  - Move new: move restrictions downwards in the game as much as possible, when there is no array reference to them.

– Simplification and elimination of collisions: Terms are simplified according to equalities that come from followings.
  - Assignments: let $x = M$ in P implies that $x = M$ in P.
  - Tests: if $M = N$ then P implies that $M = N$ in P.
  - Definitions of cryptographic primitives.
  - When a find guarantees that $x[i]$ is defined, equalities that hold at definition of x also hold under the find.
  - Elimination of collisions: if x is created by new $x : T$, $x[i] = x[j]$ implies $i = j$, up to negligible probability when T is large.

8.3 Experiment Results

CryptoVerif tested to verify the security of the some protocols such as Otway-Rees protocol that use shared-key encryption, Dnning-Sacco protocol that use public-key encryption, and Needham-Schroeder shared-key and public-key protocol. To test on these protocols, shared-key encryption is implemented as encrypt-then-MAC, using IND-CPA encryption scheme and public-key encryption is assumed to be IND-CCA2. In these settings, CryptoVerif verify secrecy of session keys and correspondence properties.

In most case, CryptoVerif succeeds in proving the desired properties when they hold, and obviously it always fails to prove them when they do not hold. However, there are a few cases in which the prover fails although the property holds. Some public-key protocols need manual proofs. Runtime of CryptoVerif is 7 mili-seconds to 35 seconds, and average is 5 seconds.

8.4 Summary

CryptoVerif is an automatic prover for the security of cryptographic protocols. The approach of CryptoVerif is to directly prove protocols in a computational model. The proof by CryptoVerif is a sequence of games described by processes. The basic proof technique is to translate games based on the observational equivalence.

CryptoVerif can verify the security of many protocols that use a variety of cryptographic primitives. Although there are a few cases in which the prover fails although the property holds, CryptoVerif succeeds in proving the desired properties and runs in a realistic time.

9 Protocol Composition Logic: Symbolic Model, Computational Model, and Applications

Speaker: John Mitchell, Stanford University

Protocol Composition Logic (PCL) is a logic for proving security properties of network protocols that use public and symmetric key cryptography. The logic is designed around a process calculus with actions for possible protocol steps including generating new random numbers, sending and receiving messages, and performing decryption and digital signature verification actions. The proof system consists of axioms about individual protocol actions and inference rules that yield assertions about protocols composed of multiple steps.

One important aspect of PCL is that the analysis is performed only considering the actions of honest parties in the protocol and the soundness of the logic permits to prove security properties of the protocol under the attack of any adversary. PCL supports compositional reasoning about complex security protocols and has been applied to a number of industry standards including SSL/TLS, IEEE 802.11i and Kerberos V5.

9.1 Motivation

Network security protocols are widely used in the real life but they are usually difficult to design and verify. Literature contains several papers that permit to prove security properties of a given protocol. In fact, they have discovered several flaws, like the ones in the Wired Equivalent Privacy (WEP) protocol as well as in the SSL (proposed) standards, 802.11i wireless authentication protocols and other. Although we can consider many of these protocols quite simple, if compared with other systems, security protocols must satisfy desired properties when an arbitrary number of sessions are executed and the attacker may use information obtained during a session to compromise other sessions.

The analysis of the correctness of a protocol can be performed in several ways, using tools like model checking, formal methods like Reactive Simulatability [3], Universally Composable framework [10] and so on. If we analyze these formal methods, we can note that they are focused on the adversary and its capability to violate the protocol.

The aim of the Protocol Composition Logic is to analyze the correctness from the point of view of the protocol, describing and reasoning on the actions performed by the honest participants of the protocol. In fact, the main goals of the PCL is to combine the advantages of the BAN logic [11] (such as to annotate programs with assertions and to have high-level direct reasoning with no explicit reasoning about the adversary) with accepted protocols semantics, for example the attacker controls the network and the protocol is described by sets of roles executed concurrently by principals.

So, consider the following Challenge-Response protocol:

$$\begin{aligned}&A \rightarrow B : N_A, A\\&B \rightarrow A : N_B, \mathit{sign}_B(N_A, N_B, A)\\&A \rightarrow B : \mathit{sign}_A(N_B, B)\end{aligned}$$

In the first message, Alice generates a nonce N_A and sends it together with her identity to Bob. Bob generates another nonce N_B and sends it to Alice together with his signature of his identity, his nonce and the nonce of Alice. Finally, Alice replies with her signature of the nonce N_B chosen by Bob and his identity. From the point of view of Alice, when she receives the message $N_B, \mathit{sign}_B(N_A, N_B, A)$, she can suppose that such message has been generated by Bob after he has received the message N_A, A sent by Alice. Similarly, Bob can suppose that Alice has sent the message $\mathit{sign}_A(N_B, B)$ after having received $N_B, \mathit{sign}_B(N_A, N_B, A)$. Moreover both Alice and Bob can be convinced that the other participant is the expected one whenever the nonces are chosen in a good way and the encryption is secure.

9.2 Description of the Logic

In order to use the PCL to study a protocol, the first thing to do is to rewrite the protocol accordingly with the syntax of the Protocol Programming Language: each protocol is a fixes set of *roles* written as a program; a *thread* is an instance of a role being executed by a principal; a single participant can execute multiple threads.

Each role is defined a sequence of actions: we have

- the *communication* actions *send* m and *receive* m;
- the *pairing* and *unpairing* actions $m := \mathit{pair}\ m_0, m_1$ and $\mathit{match}\ m\ \mathit{as}\ m_0, m_1$;
- the *encryption* and *decryption* actions $m' := \mathit{enc}\ m, k$ and $m' := \mathit{dec}\ m, k$;
- the *signature* and *verification* actions $m' := \mathit{sign}\ m, k$ and $m' := \mathit{verify}\ m, k$;
- the *nonce generation* action $\mathit{new}\ m$; and
- the *pattern matching* action $\mathit{match}\ m\ \mathit{as}\ m'$.

This means that the Challenge-Response protocol should be written as

$$
\begin{aligned}
\mathrm{Init}_{\mathrm{CR}} \equiv (A, B)[& \\
& \textit{new } m; \\
& \textit{send } A, B, < m, A >; \\
& \textit{receive } B, A, < n, \textit{sign}_B(\text{“}r\text{”}, m, n, A) >; \\
& \textit{send } A, B, \textit{sign}_A(\text{“}i\text{”}, n, B); \\
]_A &
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{Init}_{\mathrm{CR}} \equiv (B)[& \\
& \textit{receive } A, B, < m, A >; \\
& \textit{new } n; \\
& \textit{send } B, A, < n, \textit{sign}_B(\text{“}r\text{”}, m, n, A) >; \\
& \textit{receive } A, B, \textit{sign}_A(\text{“}i\text{”}, n, B); \\
]_B &
\end{aligned}
$$

Once the protocol is rewritten in the protocol programming language, it is possible to study its properties using the protocol composition logic. To do this, the logic contains the standard connectors plus some formulas that express the knowledge, the honesty and the actions performed by single agents of the protocol. In particular, there are the following formulas:

- Send(P, t): the principal P has performed an action *sendt*;
- Receive(P, t): the principal P has received the term t;
- New(P, t): the principal P has generated the new term t;
- Verify(P, t): the principal P has verified the term t;
- Decrypt(P, t): the principal P has decrypted the term t;
- Honest(P): the principal P is honest, that is it has done exactly the actions prescribed by its role;
- Fresh(P, t): the value t, generated by P, is fresh, that is no one else has seen any term containing t as subterm;
- Contains(t_1, t_2): the term t_1 contains t_2 as a subterm;
- Has(P, t): the principal P possesses information about t. This is “possess” in the limited sense of having either generated the data or received it in clear or received it under encryption where the decryption key is known. The predicate Has can be used to model secrecy properties, for example, a fact that the term t is a shared secret between threads X and Y is captured by the logical formula $\forall Z.\mathrm{Has}(Z, t) \supset (Z = X \vee Z = Y)$ where $\supset$ is the implication between formulas.

Moreover, there are modal formulas like $\phi[\textit{actions}]_P\psi$ and temporal ordering of actions $a_1 < a_2$. The modal formula $\phi[\textit{actions}]_P\psi$ represents the fact that, starting from a state where ϕ is true, the execution of actions *actions* in the thread P leads to a state that satisfies ψ; the temporal ordering $a_1 < a_2$ states that both actions a_1 and a_2 happened and a_2 has occurred after a_1.

To prove security properties of useful protocols, the PCL uses a proof system. It contains several axioms, inference rules and provides a theorem that states the correctness of formulas obtained from axioms by application of inference rules.

The core concept of the proof system is the *honesty* concept. In particular, a principal P is honest in the run R whenever the actions of X in R are precisely the interleaving of initial segments of traces of a set of roles of the protocol. Intuitively, this means that X does only what X is supposed to do. A protocol segment is a subsequence of honest party actions between pausing states. Examples of axioms are:

- $\mathit{true}\ [\mathit{send}\ m]_P\ \mathrm{Send}(P, m)$;
- $\mathrm{Honest}(P) \wedge \mathrm{Decrypt}(Y, \mathit{enc}_X(m)) \supset X = Y$;
- $\mathrm{Honest}(P) \wedge \mathrm{Verify}(Y, \mathit{sig}_X(m)) \supset \mathrm{Sign}(X, \mathit{sig}_X(m))$.

One way to prove the correctness of a protocol is to prove invariants, that is formulas that hold when threads are started and for each protocol segment, if the invariant held at the beginning of the segment, then it holds at the end. Please note that a segment is not necessary a single action: usually it contains several actions where the first one is a *receive* action and the last one is a *send*.

Once a formula ϕ is proven to be a theorem of PCL with respect to a protocol Q, then by the soundness theorem ϕ is a valid formula that implies that ϕ holds in any step of any run of Q. This means that ϕ holds for any number of participants, for each Dolev-Yao intruder and possibly for the computational model (CPCL).

The PCL supports also the protocol composition such as sequential composition of protocols or parallel composition. This means that it is possible to prove the correctness of several protocols, each one independently from the others, and then combining results to state the correctness of composed protocols.

9.3 The Computational Model

The Computational Protocol Composition Logic (CPCL) is very similar to the PCL but considers the computational aspects of the protocols. In particular, the adversary is now a probabilistic polynomial time attacker and probabilities and computational complexity are explicitly used. This means that the semantics of CPCL captures the idea that properties hold with high probability against PPT attackers.

In the computational model, one important property that is considered about a protocol is the *secretive* property: given a protocol, a nonce s and a set of keys K, the protocol is secretive if it overwhelmingly produces secretive traces, that is the thread which generates s ensures that s is encrypted with a key $k \in K$ in any message sent out and whenever a thread decrypts a message with a key $k \in K$ and parses the decryption, it ensures that the results are re-encrypted with some key $k' \in K$ in any message sent out. This notion of secretive protocols is stronger than indistinguishability: in fact, it is possible to prove that if the protocol is secretive and the nonce generator and the key holders are honest, then the key generated from the nonce satisfies indistinguishability.

The CPCL logic is essentially the same of the PCL one. The main difference is on the axioms used into the proof system:

- $\mathrm{Good}(X, a, s, K)$, if a is of an atomic type different from nonce or key

– $\text{New}(Y,n) \wedge n \neq s \supset \text{Good}(X,n,s,K)$
– $[\textit{receive } m;]_X \ \text{Good}(X,m,s,K)$
– $\text{Good}(X,m,s,K)\ [a]_X\ \text{Good}(X,m,s,K)$ for all actions a
– $\text{Good}(X,m,s,K)\ [\textit{match } m \textit{ as } m';]_X\ \text{Good}(X,m',s,K)$
– $\text{Good}(X,m_0,s,K) \wedge \text{Good}(X,m_1,s,K)\ [m := m_0.m_1]_X\ \text{Good}(X,m,s,K)$
– $\text{Good}(X,m,s,K)\ [\textit{match } m \textit{ as } m_0.m_1;]_X\ \text{Good}(X,m_0,s,K) \wedge \text{Good}(X,m_1,s,K)$
– $\text{Good}(X,m,s,K) \vee k \in K\ [m' := \textit{symencm}, k;]_X\ \text{Good}(X,m',s,K)$
– $\text{Good}(X,m,s,K) \wedge k \notin K\ [m' := \textit{symdecm}, k;]_X\ \text{Good}(X,m,s,K)$

With these axioms, the proof of the correctness of a protocol is performed in the same way of the PCL, proving invariants or deducing the “goodness” of messages exchanged during a protocol run. In particular, we prove that a protocol is secretive if it is possible to prove that each message sent out satisfies the Good formula, provided that the nonce generator is honest and that no key $k \in K$ is leaked by protocol participants.

9.4 Summary

Protocol Composition Logic (PCL) and Computational Protocol Composition Logic (CPCL) are two logics for proving security properties of network protocols that use public and symmetric key cryptography. Both logics are designed around a process calculus with actions for possible protocol steps such as generating new random numbers, exchanging of messages, and cryptographic operations. The proof system consists of axioms about individual protocol actions and inference rules that yield assertions about protocols composed of multiple steps.

Both logics provide soundness results and permit to analyze complex protocols in a compositional way, reducing the overall effort since it is possible to reuse results from previous proofs.

10 On the Use of Probabilistic Automata for Security Proofs (Part 1)

Speaker: Roberto Segala, University of Verona

Summary

In this talk, Segala devoted the first half to introduce Probabilistic I/O Automata and some interesting properties. He used it for the security proof of the cryptographic protocol. His technique is based on the hierarchical composition verification. In this technique, we make the protocol security properties simpler or smaller, we prove that the smaller protocol is correct, and we show that there is polynomially accurate simulation from the simpler protocol to the one we want to prove. In the second half of Part 1, he gave a case study. He used the technique to prove the security of Oblivious Transfer proposed by Even, Goldreich, and Lample [12]. The detail of the proof is appeared in [13].

10.1 Probabilistic I/O Automata

Probabilistic Automata is defined by

$$PA = (Q, q_0, E, H, D)$$

where Q is the set of states, $q_0 \in Q$ is the initial state, E is the set of the external actions, H is the set of the internal actions, and $D \subseteq Q \times (E \cup H) \cup Disc(Q)$ is the set of the transition relations. Segala defined $Disc(Q)$ as the set of discrete probability measure on Q. He gave an example of probabilistic automata, that is illustrated in Fig. 13.

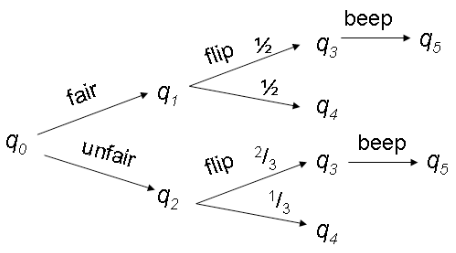


Fig. 13. Probabilistic Automata

Segala defined an *execution fragment* $\alpha = s_0 a_1 s_1 ...$ of PA which is a sequence of states and actions, such that for any i, $(s_i, a_{i+1}, \mu_{i+1} \in D$ with $\mu_{i+1}(s_{i+1}) > 0$. He also defined the sequence $s_0 s_1 s_2 ...$ as a trace.

Next, he discussed about the composition of two probabilistic automata. Let $\mathcal{A}_1 = (Q_1, q_1, E_1, H_1, D_1)$ and $\mathcal{A}_2 = (Q_2, q_2, E_2, H_2, D_2$ be two probabilistic automata. The composition result $\mathcal{A}_1 || \mathcal{A}_2$ is $(Q_1 \times Q_2, (q_1, q_2), E_1 \cup E_2, H_1 \cup H_2, D)$ where

$$D = \{(q, a, \mu_1 \times \mu_2) | (\pi_i(q), a, \mu_i) \in D_i \text{ if } a \in E_i \cup H_i, \mu_i = \delta(\pi_i(a)) \text{ otherwise}\}.$$

Segala presented an example of the composition of probabilistic automata that is shown in Fig. 14.

Segala also defined the projection, which is the inversion of the composition. Let the execution fragment α of probabilistic automata $\mathcal{A}_1 || \mathcal{A}_2$ be

$$(q_0, s_0)\mathbf{d}(q_1, s_1)\mathbf{ch}(q_3, s_1)\mathbf{coffee}(q_5, s_3).$$

The projection of α on $\mathcal{A}_1$, $\pi_1(\alpha)$ is $q_0\mathbf{d}q_2\mathbf{ch}q_3\mathbf{coffee}q_5$. On the other hand, the projection on $\mathcal{A}_2$, $\pi_2(\alpha)$, is $s_0\mathbf{d}s_1\mathbf{coffee}s_3$.

As the last part of the first half of the lecture, Segala defined the forward simulations of two probabilistic automata. According to [14], the simulation from a PA $\mathcal{A}_1$ to PA $\mathcal{A}_2$ is a relation $\mathcal{R}$ from Q_1 to Q_2 such that

– $q_1 \mathcal{R} q_2$

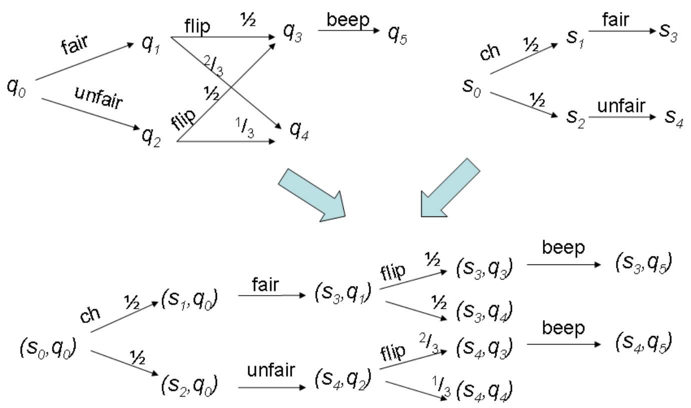


Fig. 14. The Composition of Probabilistic Automata

– for each pair $(q_i, s_i) \in \mathcal{R}$, if $(q_i, a, \mu_1) \in D_1$, then there exists $(s_i, a, \mu_2) \in D_2$ such that $\mu_1 \mathcal{L}(\mathcal{R}) \mu_2$.

Segala defined the lifting $\mathcal{L}(\mathcal{R})$ of a relation $\mathcal{R}$. $\mathcal{L}(\mathcal{R})$ is a relation from $Disc(Q_1)$ to $Disc(Q_2)$ such that $\rho_1\ \mathcal{L}(\mathcal{R})\ \rho_2$ if and only if there exists a weighting function $w\colon Q_1 \times Q_2 \to [0, 1]$ such that

– $w(q_1, q_2) > 0$ implies $q_1\ \mathcal{R}\ q_2$
– $\sum_{q_1} w(q_1, q_2) = \rho_2(q_2)$
– $\sum_{q_2} w(q_1, q_2) = \rho_1(q_1)$

If the relation $\mathcal{R}$ exists, we say that $\mathcal{A}_1$ is simulated by $\mathcal{A}_2$ or $\mathcal{A}_1 \leq \mathcal{A}_2$.

10.2 Oblivious Transfer (OT)

To describe the OT problem, we refer to [13]. According to this paper, two input bits $(x(0), x(1))$ are submitted to a transmitter and a single input bit i to a receiver. The expectation of the protocol is that the receiver is able to tell $x(i)$ without knowing which is the bit $x(1 - i)$. Moreover, we do not want the transmitter to know i.

The protocol proposed by Even, Goldreich, and Lampel [12] is illustrated in Fig. 15. Please note that f is a random trap-door permutation selected by the transmitter and B is a hardcore-predicate for f.

To prove the correctness of the protocol using probabilistic automata, Segala introduced six PA's systems defined by $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6$. $\mathcal{A}_1$ represents the real system which is the composition of two automata representing the protocol parties and an automaton representing an adversarial communication service. In this case, the adversary has access to all messages sent during execution of protocol. $\mathcal{A}_6$ represents the ideal system which is the composition of an ideal oblivious transfer functionality automaton specifying the allowed input/output

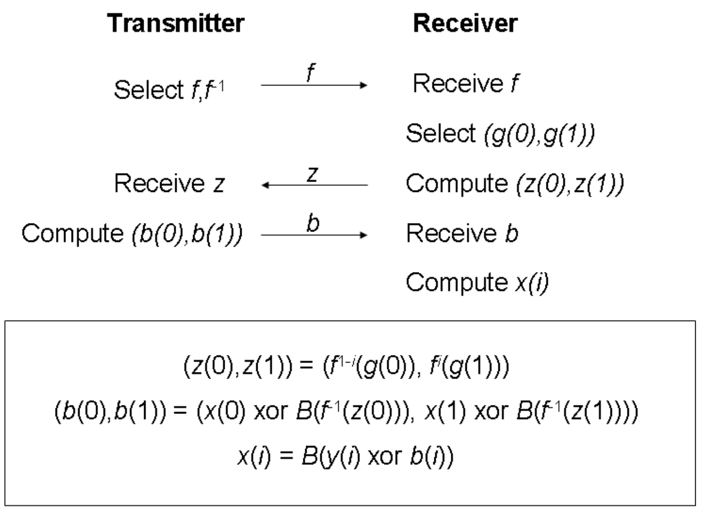


Fig. 15. The Protocol for Oblivious Transfer

behavior for oblivious transfer, and a simulator automaton interacting with the functionality and trying to mimic the behavior of the real system. The other systems $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$, $\mathcal{A}_5$ are the intermediate systems, which are used to obtain an easier proof.

The security of the ideal system $\mathcal{A}_6$ is not a difficult task to be proved. In this work, Segala showed that $\mathcal{A}_1 \leq_0 \mathcal{A}_2 \leq_0 \mathcal{A}_3 \leq_{neq,pt} \mathcal{A}_4 \leq_0 \mathcal{A}_5 \leq_0 \mathcal{A}_6$. We can infer that $\mathcal{A}_1 \leq_{neq,pt} \mathcal{A}_6$ which means that the real system can be simulated by the ideal system, and the simulation result *computationally.* The word *computationally* means that the real and the ideal system cannot be distinguished by a polynomial-time-bounded observer.

11 On the Use of Probabilistic Automata for Security Proofs (Part 2)

Speaker: Roberto Segala, University of Verona

Summary

In the first part, Segala gave the foundation of probabilistic automata, and one case study on the oblivious transfer protocol. On the second part, he gave two more case studies. The first one is the correctness proof of message authentication protocol proposed by Bellare and Rogaway [15], and the second case study is the improvement of the work by Cortier and Warinschi [16] on Dolev-Yao soundness.

$$\begin{tikzcd}[column sep=huge]
A \arrow[r, shift left=6, "R_A"] \arrow[r, leftarrow, "{[b.a.R_A.R_B]_s}"] \arrow[r, shift right=6, "{[a.R_B]_s}"] & B
\end{tikzcd}$$

Fig. 16. MAP1 Protocol

11.1 Agent Authentication

Let all players belong to a group $\mathbb{A}$ which all members of the group share a secret code s and a pseudorandom function. The event that a player $A \in \mathbb{A}$ wants to prove its identity with player B is called as *agent authentication*, and the message for agent authentication is called as a message authentication code (MAC). The protocol is proposed by Bellare and Rogaway [15], and is called MAP1. We illustrate MAP1 in Fig. 16. Note that A and B are descriptions of the identity of A and B, respectively, and R_A, R_B are nonces generated by A and B, respectively.

In the paper proving the correctness of this protocol [14], authors refer to the approximate simulations. Approximate simulation is a simulation that permits some error ε to occur. To define it formally, we refer to lifting function $\mathcal{L}(\mathcal{R})$ defined in the first part of this report. Here, Segala defined $\mathcal{L}(\mathcal{R}, \varepsilon)$:

1. if $\varepsilon > 1$, then $\rho_x\ \mathcal{L}(\mathcal{R}, \varepsilon)\ \rho_y$.
2. if $\varepsilon \in [0, 1]$, then $\rho_x\ \mathcal{L}(\mathcal{R}, \varepsilon)\ \rho_y$ if there exist $\rho'_x, \rho''_x \in \mathit{Disc}(X)$ and $\rho'_y, \rho''_y \in \mathit{Disc}(Y)$ such that
   - $\rho_x = (1 - \varepsilon)\rho'_x + \varepsilon\rho''_x$,
   - $\rho_y = (1 - \varepsilon)\rho'_y + \varepsilon\rho''_y$, and
   - $\rho'_x\ \mathcal{L}(\mathcal{R})\ \rho'_y$.

For instance, let we have $p(k)$ nonces $n_1, ..., n_{p(k)}$ when $p(k)$ is any polynomial of k, then $\Pr[n_i = n_j | i \neq j] < k^{-c}$ in the situation that we choose nonces randomly. This make the automata system using randomly-chosen nonces and ideally-chosen nonces differ by the probability $k^{-c} = \varepsilon$ on each step. The upper bound of the error is $p(k)\varepsilon = p(k)k^{-c}$ for the polynomial-bounded probabilistic automata.

Similar to the correctness proof of oblivious transfer in Part 1, they prove the correctness of the protocol using the hierarchical compositional verification. In this case study, there are three layers illustrated in Fig. 17. Note that NG^R is a randomly-chosen nonce generator, NG^I is an ideal nonce generator, $RAdv^f$ is an adversary controlled by a generic probabilistic polynomial time algorithm f, $GAdv$ is a nondeterministic automaton that is permitted to perform any action except for casting new message authentication codes without obtaining them from the agents, and A_i are players in message authentication protocol.

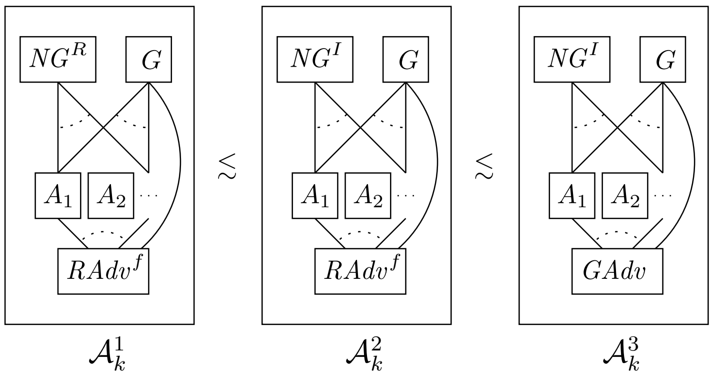


Fig. 17. The three layers used for proof of MAP1 using PA

11.2 Dolev-Yao Soundness

In the last part of the lecture, Segala referred to the paper published by Cortier and Warinschi [16] on computational soundness. It is the task to bridge the symbolic proof in Dolev-Yao model and the computational model. In this work, Cortier and Warinschi have contributed on the soundness of protocols that use signatures and asymmetric encryption.

Segala referred to the protocol syntax and the execution models defined in the paper. He used the probabilistic automata and including random bits with a probabilistic measure in the proof. This makes the proof in the crucial part easier. The hierarchical compositional verification, that is applied to oblivious transfer and agent authentication, is also applied in this case.

References

1. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology **15**(2) (2002) 103–127
2. Abadi, M., Warinschi, B.: Security analysis of cryptographically controlled access to XML documents. Journal of the ACM **55**(2) (2008) 1–29
3. Backes, M., Pfitzmann, B., Waidner, M.: A composable cryptographic library with nested operations (extended abstract). In: CCS '03: Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM Press (2003) 220–230
4. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6) (1976) 644–654
5. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **2**(29) (1983)
6. Bellare, M., Rogaway, P.: Optimal asymmetric encryption - how to encrypt with rsa. In: Eurocrypt'94. Volume 950 of Lecture Notes in Computer Science. (1995) 341–358
7. Boneh, D., Franklin, M.: Identity based encryption from the weil pairing. SIAM Journal of Computing **32**(3) (2003) 586–615

8. Backes, M., Pfitzmann, B., Waidner, M.: A general composition theorem for secure reactive systems. In: Theory of Cryptography (TCC 2004). Volume 2951 of Lecture Notes in Computer Science., Springer Verlag Heidelberg (2004) 336–354
9. Datta, A., Küsters, R., Mitchell, J.C., Ramanathan, A.: On the relationships between notions of simulation-based security. In: Theory of Cryptography. Volume 3378 of Lecture Notes in Computer Science., Springer Berlin Heidelberg (2005) 476–494
10. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: The 42nd Annual Symposium on Foundations of Computer Science. (2001) 136–145
11. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Transactions on Computer Systems (TOCS) **8**(1) (1990) 18–36
12. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contrats. Communications of the ACM **28**(6) (1985) 637–647
13. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Analyzing security protocols using time-bounded task-PIOAs. Discrete Event Dynamic Systems **18**(1) (2008) 111–159
14. Segala, R., Turrini, A.: Approximated computationally bounded simulation relations for probabilistic automata. In: Proceedings, 20th IEEE Computer Security Foundations Symposium, IEEE Computer Society Press (2007) 140–154
15. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, Springer Verlag New York, Inc (1994) 232–249
16. Cortier, V., Warinschi, B.: Computationally sound, automated proofs for security protocols. In: Programming Languages and Systems. Volume 3444 of Lecture Notes in Computer Science., Springer Berlin Heidelberg (2005) 157–171
17. Micciancio, D., Warinschi, B.: Soundness of formal encryption in the presence of active adversaries. In: Theory of Cryptography Conference – TCC'04. Volume 2951 of Lecture Notes in Computer Science., Springer Verlag Heidelberg (2004) 133–151