High-Level Programming for E-Cash

Pedro Adão
IT and IST, Lisboa

Francesco Zappa Nardelli
INRIA and MSR-INRIA Joint Centre

Nataliya Guts
MSR-INRIA Joint Centre
E-Cash Protocols

- Introduced by Chaum in 1982 it intends to simulate the use of traditional money
  - Many interesting properties of the traditional money are delicate to mimic in digital world

- **Aim** to provide robust abstractions for anonymous payment protocols
  - Users should spend coins anonymously
  - Users cannot forge coins
  - User should not spend the same coin twice without being eventually caught
  - Spending should be offline

- By necessity, these protocols involve sophisticated cryptographic constructions
E-Cash Protocol Spec

- **Withdraw(\(U(pk_\mathcal{B}, sk_\mathcal{U}, n)\), \(B(pk_\mathcal{U}, sk_\mathcal{B}, n)\))** protocol, the user \(U\) withdraws a wallet \(W\) of \(n\) coins from the bank \(B\). The users output is the wallet \(W\), or an error message. \(B\)'s output is some information \(T_W\) which will allow the bank to trace the user should this user double-spend some coin, or an error message. The bank maintains a database \(D\) for this trace information, to which it enters the record \((pk_\mathcal{U}, T_W)\).

- **Spend(\(U(W, pk_\mathcal{M})\), \(M(sk_\mathcal{M}, pk_\mathcal{B}, n)\))** protocol, a user \(U\) gives one of the coins from his wallet \(W\) to the merchant \(M\). Here, the merchant obtains a serial number \(S\) of the coin, and a proof \(\pi\) of validity of the coin. The users output is an updated wallet \(W'\).

- **Deposit(\(M(sk_\mathcal{M}, S, \pi, pk_\mathcal{B})\), \(B(pk_\mathcal{M}, sk_\mathcal{B})\))** protocol, a merchant \(M\) deposits a coin \((S, \pi)\) into its account held by the bank \(B\). Whenever an honest \(M\) obtained \((S, \pi)\) by running the Spend protocol with any (honest or otherwise) user, there is a guarantee that this coin will be accepted by the bank. \(B\) adds \((S, \pi)\) to its list \(L\) of spent coins. The merchants output is nothing or an error message.
E-Cash Protocol Spec

- **Identify**$(params, S, π_1, π_2)$ algorithm allows to identify double-spenders using a serial number $S$ and two proofs of validity of this coin, $π_1$ and $π_2$, possibly submitted by malicious merchants. This algorithm outputs a public key $pk_U$ and a proof $Π_G$. If the merchants who had submitted $π_1$ and $π_2$ are not malicious, then $Π_G$ is evidence that $pk_U$ is the registered public key of a user that double-spent coin $S$.

- **VerifyGuilt**$(params, S, pk_U, Π_G)$ algorithm allows to publicly verify proof $Π_G$ that the user with public key $pk_U$ is guilty of double-spending coin $S$. 
Correctness
Balance
Identification of Double Spender

- Scalf: 1150
- Scarf: 1124
- Xetafe: 1576
- Xetafee: 1557/1579
- Xelatfe: 1590:1603
- Getaphe: 1714
- Gelate: 1775
Tracing of Double Spender

- Scared
- Scarf 1124
- Xeafle 1576
- Xeafsee 1557/1579
- Xeafsee 1590/1603
- Getophe 1714
- Gelate 1775
Anonymity of the User
Withdraw($U(pk_B, sk_U, 2^\ell), B(pk_U, sk_B, 2^\ell)$): A user $U$ interacts with the bank $B$ as follows:

1. $U$ identifies himself to the bank $B$ by proving knowledge of $sk_U$.
2. In this step, the user and bank contribute randomness to the wallet secret $s$; the user also selects a wallet secret $t$. This is done as follows: $U$ selects random values $s', t \in \mathbb{Z}_q$ and sends a commitment $A' = \text{PedCom}(u, s', t; r)$ to $B$. $B$ sends a random $r' \in \mathbb{Z}_q$. Then $U$ sets $s = s' + r'$. $U$ and $B$ locally compute $A = g_2^{s'} A' = \text{PedCom}(u, s' + r', t; r) = \text{PedCom}(u, s, t; r)$.
3. $U$ and $B$ run the CL protocol for obtaining $B$’s signature on committed values contained in commitment $A$. As a result, $U$ obtains $\sigma_B(u, s, t)$.
4. $U$ saves the wallet $W = (sk_U, s, t, \sigma_B(u, s, t), J)$, where $s, t$ are the wallet secrets, $\sigma_B(u, s, t)$ is the bank’s signature, and $J$ is an $\ell$-bit coin counter initialized to zero.
5. $B$ records a debit of $2^\ell$ coins for account $pk_U$. 
An E-Cash Protocol [CHL05]

Spend($\mathcal{U}(W, pk_\mathcal{M}), \mathcal{M}(sk_\mathcal{M}, pk_B, 2^\ell)$): $\mathcal{U}$ anonymously transfers a coin to $\mathcal{M}$ as follows. (An optimized version appears in the full version of this paper [13].)

1. $\mathcal{M}$ (optionally) sends a string $info \in \{0, 1\}^*$ containing transaction information to $\mathcal{U}$ and authenticates himself by proving knowledge of $sk_\mathcal{M}$.

2. $\mathcal{M}$ chooses a random $R \in \mathbb{Z}_q^*$ and sends $R$ to $\mathcal{U}$. This is for the double-spending equation (see Section 1).

3. $\mathcal{U}$ sends to $\mathcal{M}$ the serial number of the coin $S = F_{s}^{DY}(J)$, and security tag $T = pk_\mathcal{U}F_{t}^{DY}(J)^R$. Now $\mathcal{U}$ must prove their validity, i.e., that $S$ and $T$ correspond to wallet secrets $(u, s, t)$ signed by $\mathcal{B}$. This is done as follows:
   (a) Let $A = \text{PedCom}(J)$; prove that $A$ is a commitment to an integer in the range $[0 \ldots 2^\ell - 1]$.
   (b) Let $B = \text{PedCom}(u)$, $C = \text{PedCom}(s)$, $D = \text{PedCom}(t)$; prove knowledge of a CL signature from $\mathcal{B}$ on the openings of $B, C$ and $D$ in that order.
   (c) Prove $S = F_{s}^{DY}(J) = g_2^{1/(J+s+1)}$ and $T = pk_\mathcal{U}F_{t}^{DY}(J)^R = g_2^{u+R/(J+t+1)}$. More formally, this proof is the following proof of knowledge:

   \[
   PK\{ (\alpha, \beta, \delta, \gamma_1, \ldots, \gamma_3) : g_1 = (AC)^\alpha h_1^{\gamma_1} \land S = g_2^\alpha \land g_1 = (AD)^\beta h_1^{\gamma_2} \land B = g_1^\delta h_1^{\gamma_3} \land T = g_2^\delta (g_2^R)^\beta \} \]

   Use the Fiat-Shamir heuristic to turn all the proofs above into one signature of knowledge on the values $(S, T, A, B, C, D, g_1, h_1, n, g_2, pk_\mathcal{M}, R, info)$. Call the resulting signature $\Phi$.

4. If $\Phi$ verifies, $\mathcal{M}$ accepts the coin $(S, \pi)$, where $\pi = (R, T, \Phi)$, and uses this information at deposit time.

5. $\mathcal{U}$ updates his counter $J = J + 1$. When $J > 2^\ell - 1$, the wallet is empty.
Problem

- How to symbolically model E-Cash Protocols?
- How to reason about it?
  - E-Cash properties involve elaborated cryptographic statements
- And what about reasoning on E-Cash at an application layer?
  - Do we need to redo all these proofs?
  - How is the interaction of programs and crypto?
- Are we willing to model every single cryptographic or do we rather prefer to model E-Cash primitives as BB?
The 3 Layer Cake

- **Well-behaved Semantics**: Users are modelled as applied pi-calculus processes.
  - All users follow the specification.
  - Double spending is prevented by construction.
- **Intermediate Semantics**: Environment is an arbitrary process.
- **Crypto**:
This Work

- We consider symbolic characterizations of Compact E-Cash protocols following the specification proposed by Camenisch, Hohenberger, and Lysyanskaya [CHL05].

- We design and implement a distributed (asynchronous) process calculus with high-level E-Cash primitives and communication (following ideas of [AF06]).
  - Our calculus supports simple reasoning, based on labelled transitions and observational equivalence.

- We consider two variants of the symbolic semantics:
  - An abstract semantics that excludes any double spending (by design).
  - A more realistic intermediate semantics that accounts for the possibility of double spending (with reliable detection).
  - We show that any trace of the intermediate semantics can be captured by the “honest” semantics or an alert is issued.
This Work

- We then consider a direct cryptographic implementation of high-level E-Cash primitives (Withdraw, Spend and Deposit following ideas of [AF06])

- Relate the intermediate semantics to the computational properties of the underlying E-cash protocol

- We obtain soundness and completeness for processes, in the presence of active adversaries

- We do not rely on DY abstractions of cryptographic primitives
  - Full abstraction for spi or applied pi calculus is too hard
High-Level Processes
Processes and Systems

\[ P ::= \]

0 \quad \text{null process}

P_1 \mid P_2 \quad \text{deterministic parallel composition}

\nu \ c \ . \ P \quad \text{restriction of names}

u?(x).P \quad \text{receive on } u

u!(M).P \quad \text{output } M \text{ on } u, M \text{ is not a channel name}

\text{if } M = M' \text{ then } P \text{ else } P' \quad \text{conditional}

\text{repl } P \quad \text{replication of a blocking process } P

\text{withdraw! } u \ L \quad \text{withdraw a coin from bank } u

\text{withdraw? } u \ P \quad \text{create a coin for user } u

\text{spend! } u \ u' \ p \quad \text{spend the coin } u' \text{ to pay } u \text{ through bank } p

\text{spend? } p \ L \quad \text{wait a payment through bank } p

\text{deposit! } p \ u \ u' \quad \text{deposit a spent coin } u \text{ to bank } p \text{ (user } u' \text{ is hidden)}

\text{deposit? } p \ (x) \ P_1 \ P_2 \quad \text{bind a coin, and its depositor in } P_1 \text{ if it is honest}

A, E ::= 

0 \quad \text{process owned by } p

p@P \quad \text{parallel composition}

A_1 \mid A_2 \quad \text{restriction of name}
Reduction Semantics

\[ U @\text{withdraw}! B \mathcal{L} \mid B @\text{withdraw}? U P \rightarrow_a \nu b . ( U @\mathcal{L}[\text{spend}! x b B] \mid B @ P \mid \text{coin } b B ) \]

\[ U @\text{spend}! M b B \mid M @\text{spend}? B \mathcal{L} \rightarrow_a M @\mathcal{L}[\text{deposit}! B b] \]

\[ M @\text{deposit}! B b \mid B @\text{deposit}? M (x) P_1 P_2 \mid \text{coin } b B \rightarrow_a B @ P_1 \{ M / x \} \]
Labelled Transition Semantics

\[
\begin{align*}
A & \rightarrow A' & \text{AbsLtSilent} \\
\frac{A \overset{\tau}{\rightarrow} A'}{A \rightarrow A'}
\end{align*}
\]

\[
\begin{align*}
a@c!(\langle M \rangle).P & \xrightarrow{c!M} a@P & \text{AbsLtSend_term} \\
\frac{a@c!\langle M \rangle.P \xrightarrow{c!M} a@P}{a@c?(x).P \xrightarrow{c?M} a@P\{^M_x\}} & \text{AbsLtReceive_term}
\end{align*}
\]

\[
\begin{align*}
A & \xrightarrow{c!M} A' \\
\frac{(c \neq r \land r \in M)}{\nu r.A \xrightarrow{\nu r.c!M} A'} & \text{AbsLtOpen}
\end{align*}
\]

\[
\begin{align*}
A & \xrightarrow{\phi} A' \land c \notin \phi & \text{AbsLtScope} \\
\frac{A \xrightarrow{\phi} A' \land c \notin \phi}{\nu c.A \xrightarrow{\phi} \nu c.A'}
\end{align*}
\]

\[
\begin{align*}
A_1 \xrightarrow{\phi} A_1' \land BN(\phi) \cap FN(A_2) = \emptyset \\
\frac{A_1 \mid A_2 \xrightarrow{\phi} A_1' \mid A_2}{\text{AbsLtPar}}
\end{align*}
\]

\[
\begin{align*}
A_1 \equiv A_2 \land (A_2 \xrightarrow{\phi} A_3 \land A_3 \equiv A_4) \\
\frac{A_1 \equiv A_2 \land (A_2 \xrightarrow{\phi} A_3 \land A_3 \equiv A_4)}{A_1 \xrightarrow{\phi} A_4} & \text{AbsLtLab_struct}
\end{align*}
\]
Labelled Transition Semantics

\[ U \@ \text{withdraw!} \, B \, \mathcal{L} \quad \xrightarrow{\text{withdraw! } U \, B} \quad \nu b \, . \, U \@ \mathcal{L}[\text{spend! } x \, b \, B] \]

\[ B \@ \text{withdraw? } U \, P \quad \xrightarrow{\text{withdraw? } U \, B} \quad B \@ P \quad | \quad \text{coin } b \, B \]

\[ U \@ \text{spend! } M \, b \, B \quad \xrightarrow{\text{spend! } b \, B \, M} \quad U \@ 0 \]

\[ M \@ \text{spend? } B \, \mathcal{L} \quad | \quad \text{coin } b \, B \quad \xrightarrow{\text{spend? } b \, B \, M} \quad M \@ \mathcal{L}[\text{deposit! } B \, b] \quad | \quad \text{coin } b \, B \quad \quad (B \in \mathcal{H}) \]

\[ M \@ \text{spend? } B \, \mathcal{L} \quad \xrightarrow{\text{spend? } b \, B \, M} \quad M \@ \mathcal{L}[\text{deposit! } B \, b] \quad \quad (B \notin \mathcal{H}) \]

\[ M \@ \text{deposit! } B \, b \quad \xrightarrow{\text{deposit! } b \, B \, M} \quad M \@ 0 \]

\[ B \@ \text{deposit? } M \,(x) \, P_1 \, P_2 \quad | \quad \text{coin } b \, B \quad \xrightarrow{\text{deposit? } b \, B \, M} \quad B \@ P_1 \{M/x\} \]
Intermediate Reduction Semantics

\[ U \odot \text{withdraw!} \ B \mathcal{L} \ | \ B \odot \text{withdraw?} \ U \ P \rightarrow i \nu b \ (U \odot \mathcal{L}[\text{spend!} \ x \ b \ B] \ | \ B \odot P \ | \ \text{coin} \ b \ B \ U) \]

\[ U \odot \text{spend!} \ M \ b \ B \ | \ M \odot \text{spend?} \ B \ \mathcal{L} \rightarrow i \ M \odot \mathcal{L}[\text{deposit!} \ B \ b] \]

\[ M \odot \text{deposit!} \ B \ b \ | \ B \odot \text{deposit?} \ M (x) P_1 P_2 \ | \ \text{coin} \ b \ B \ U \rightarrow i \ B \odot P_1 \{^M/x\} \ | \ \text{coin}^d \ b \ B \ U \]

\[ M \odot \text{deposit!} \ B \ b \ | \ B \odot \text{deposit?} \ M (x) P_1 P_2 \ | \ \text{coin}^d \ b \ B \ U \rightarrow i \ B \odot P_2 \{^\text{bad}(b, x)/x\} \ | \ \text{coin}^d \ b \ B \ U \]
Intermediate LT Semantics

\[
\begin{align*}
U \mathbin{@} \text{withdraw}! & B \not\mathbin{\text{L}} \quad \text{withdraw}! \quad U \not\mathbin{\text{B}} \quad \xrightarrow{\nu b} \quad U \mathbin{@} \mathbin{\text{L}}[\text{spend}! \ x \ b \ B] \\
B \mathbin{@} \text{withdraw}? & U \not\mathbin{\text{P}} \quad \text{withdraw}? \quad U \not\mathbin{\text{B}} \quad \xrightarrow{i} \quad B \mathbin{@} \mathbin{\text{P}} \quad \vert \quad \text{coin} \ b \ B \ U
\end{align*}
\]

\[
\begin{align*}
U \mathbin{@} \text{spend}! & M \ b \ B \quad \text{spend}! \ b \ B \ M \quad \xrightarrow{i} \quad U \mathbin{@} 0 \\
M \mathbin{@} \text{spend}? & B \not\mathbin{\text{L}} \quad \text{coin}^* \ b \ B \ U \quad \text{spend}? \ b \ B \ M \quad \xrightarrow{i} \quad M \mathbin{@} \mathbin{\text{L}}[\text{deposit}! \ B \ b] \quad \text{coin}^* \ b \ B \ U \quad (B \in \mathcal{H}) \\
M \mathbin{@} \text{spend}? & B \not\mathbin{\text{L}} \quad \text{spend}? \ b \ B \ M \ U \quad \xrightarrow{i} \quad M \mathbin{@} \mathbin{\text{L}}[\text{deposit}! \ B \ b] \quad (B \notin \mathcal{H})
\end{align*}
\]

\[
\begin{align*}
M \mathbin{@} \text{deposit}! & B \ b \quad \text{deposit}! \ b \ B \ M \quad \xrightarrow{i} \quad M \mathbin{@} 0 \\
B \mathbin{@} \text{deposit}? & M \ (x) \ P_1 \ P_2 \quad \text{coin}^d \ b \ B \ U \quad \text{deposit}? \ b \ B \ M \quad \xrightarrow{i} \quad B \mathbin{@} P_1 \{M/_{x}\} \quad \text{coin}^d \ b \ B \ U \\
B \mathbin{@} \text{deposit}? & M \ (x) \ P_1 \ P_2 \quad \text{coin}^d \ b \ B \ U \quad \text{deposit}? \ b \ B \ M \quad \xrightarrow{i} \quad B \mathbin{@} P_2 \{\text{bad}(b, U)/_{x}\} \quad \text{coin}^d \ b \ B \ U
\end{align*}
\]
High-Level Reasoning
**Example – Properties**

- **Correctness.** An honest user can withdraw a coin from an honest bank, according to **Withdraw**.
  
  Only the user specified by the bank can receive the spend continuation. Only the merchant specified by the client can receive the deposit continuation.

- **Balance.** For any series of reductions of an initial high level configuration, rule **DEPOSIT** cannot be applied more often than **Withdraw**.

- **Anonymity of users.** A bank cannot learn anything about a user’s spendings

\[
p_1@\text{spend! } b \ S \ B \mid p_2@0 \cong p_1@0 \mid p_2@\text{spend! } b \ S \ B \quad \forall p_1, p_2
\]
Example – Properties

- *Identification of double-spenders.* Given two records of a double spent coin, the identity of the user can be extracted.

- *Weak Exculpability.* Only double-spenders can be accused.

- *Strong Exculpability.* An user can only be responsible for coins that he indeed double-spent (each $\text{bad}(b, U)$ binds user $U$ to a particular coin $b$).
Soundness, Completeness for High-Level Semantics
Main Results

Well-behaved Semantics

Intermediate Semantics

Crypto

Claim 1 (Relative correctness) If two high level processes are equivalent then they are also equivalent in the weak intermediate semantics, if $A \sim A'$ then $A \sim_w A'$.

Claim 2 (Completeness) If two well-formed high level processes are equivalent in the intermediate level semantics then they are also equivalent in the high level semantics, if $A \sim_i A'$ then $A \sim A'$.

Claim 3 (Progress) If $P$ is cheated at bank $B$ on coin $b$, and if $B$ is honest, then spender of the coin will be accused by $B$. 

Low-Level Target Model
Low-Level Execution Model

- Systems consist of a finite number of communicating principals that are complying with the HL semantics.
  - THEY MAY DOUBLE SPEND

- Each principal runs its own program within a PPT machine.
  - includes 3 crypto boxes to perform the 3 ecash protocols
  - has an input and an output tape

- Priority is given to honest users

- Whenever all machines complete, all messages to the adversary are shuffled, and given to the adversary
Low-Level Execution Model

- When activated, a machine reads one message from its input tape
  - an ecash message is routed to the appropriate cryptobox
  - a communication message, is sent to the running process

- Ecash primitives in evaluation context
  - An output primitive triggers a new session of the protocol
  - An input primitive starts a waiting thread

- All machines should run to completion
  - consume all messages in its input tape and
  - write all output messages in the input tapes of receivers
Low-Level Equivalence (Target)

Two PPT systems $M^0$ and $M^1$ are equivalent, written $M^0 \approx M^1$, when for every PPT adversary $A$, we have $|\Pr[1 \leftarrow A[M^0_\eta]] - \Pr[1 \leftarrow A[M^1_\eta]]| \leq \text{neg } (\eta)$.
Soundness, Completeness wrt Computational Cryptography
Main Results

**Theorem 1 (Correctness)** For all systems $A$ with a single evaluation context and $D$ if $A \xrightarrow[]{\phi} A'$ then $\exists A, \tilde{\sigma}, D'$ such that $A[M(A, D)] \xrightarrow{\tilde{\sigma}} M(A', D')$.

**Theorem 2 (Completeness)** If the implementations of two intermediate processes are equivalent then those processes are also equivalent.

**Claim 3 (Trace Lifting)** For all systems $A$ and $D$, if $M(A, D) \xrightarrow{\sigma_1, \sigma_2} M'$ then $\exists A, D'$ such that $A \xrightarrow{\phi_i} A'$ and $M' = M(A', D')$.

**Claim 4 (Soundness)** If two intermediate processes are equivalent then with an overwhelming probability their low level implementations are also equivalent.

Well-behaved Semantics

Intermediate Semantics

Crypto
Reasoning about cryptographic actions is much more fun than cryptographic terms 😊

We define a 3-layer cake to reason about E-Cash protocols
- A low Crypto layer
- An intermediate symbolic layer where probabilities are discarded
- A higher symbolic layer where bad behaviours do not occur (by construction)

We “show” that the low-level crypto layer is correctly abstracted by the “well-behaved” semantics

Formalizing properties of E-Cash protocols helped us understand and fix part of the specification
Future Work

- Since it is work in progress....
- Try to debug and clarify specification
- Try to adapt similar techniques to other instances
  - Eg, e-voting
High-Level Programming for E-Cash

Pedro Adão
IT and IST, Lisboa

Cédric Fournet
Microsoft Research and MSR-INRIA Joint Centre

Nataliya Guts
MSR-INRIA Joint Centre

Francesco Zappa Nardelli
INRIA and MSR-INRIA Joint Centre