Divisible E-cash Made Practical

Sébastien Canard[1], David Pointcheval[2], Olivier Sanders[1,2] and Jacques Traoré[1]

(1) Orange Labs, Caen, France
(2) École Normale Supérieure, CNRS & INRIA, Paris, France

PKC 2015, March 30, 2015

Agenda

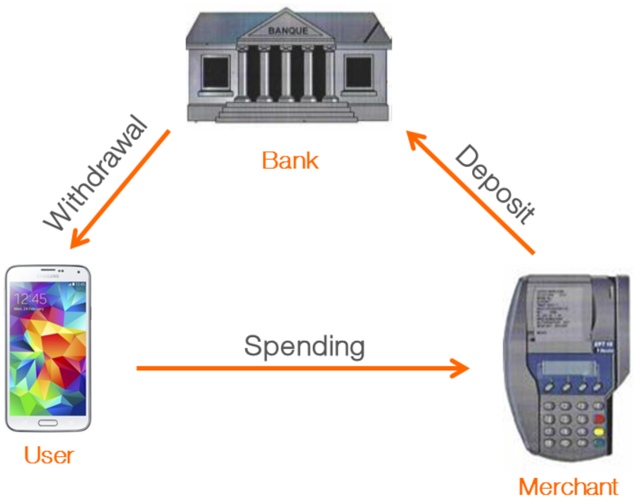
- E-Cash
- Related Works
- Our construction
- Achieving Anonymity
- Divisible E-cash Made Practical?
- Conclusion

E-Cash

Context

- Electronic payment systems offer greater convenience to end-users but at the cost of a loss in terms of privacy
- In 1982, Chaum proposed E-cash to reconcile the benefits of both solutions
- E-cash is the digital analogue of regular money

E-Cash



Security Properties

- Users must be anonymous
- Banks must be able to detect double spendings
- Defrauders must be identified
- The detection should be performed offline

Divisible E-cash

- Users of E-cash systems spend coins one by one
- To remain efficient, one must use several denominations

  $\Longrightarrow$ cumbersome for users

  $\Longrightarrow$ change issues
- Divisible E-cash Systems allow users to withdraw a coin of value V and to spend parts of it efficiently

Anonymity

Different notions of anonymity:

- Weak Anonymity: transactions involving the same coin are linkable
- Unlinkability: transactions involving the same coin are unlinkable but some information on the coin is revealed
- Strong Unlinkability: transactions involving the same coin are unlinkable and no information on the coin is revealed
- Anonymity: identification of defrauders can be performed without a trusted entity

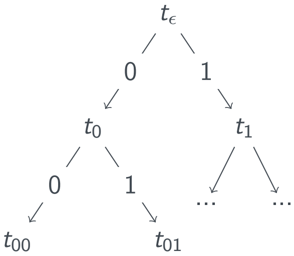
Related Works

Related Works

- Eurocrypt 2007: Achieving anonymity is possible. Unpractical construction in the ROM
- FC 2008: More efficient construction but unconventional security model
- FC 2010: Improvement of the construction of EC 07. Still too complex
- Pairing 2012: Unpractical construction in the standard model

Divisible coin

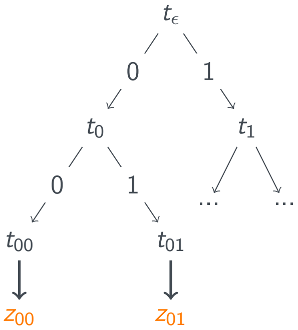
A coin of value 2^n is associated with a binary tree of depth n



Every node s is associated with an element t_s

Divisible coin

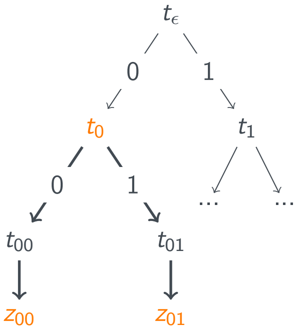
A coin of value 2^n is associated with a binary tree of depth n



Every leaf f is associated with a serial number z_f

Divisible coin

A coin of value 2^n is associated with a binary tree of depth n



Given t_s we can recover z_f for every leaf f descending from s

Security

- To spend a value 2^l, the user reveals t_s with s of depth $n - l$

  $\Longrightarrow$ implicitly reveals 2^l serial numbers

- Revealing t_s must not leak any information on the other serial numbers.

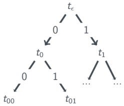
- Only 2^n serial numbers by coin

  $\Longrightarrow$ double spendings can be detected

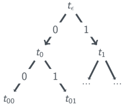
- Divisible E-cash systems without serial numbers are unpractical

Previous Constructions

Alice | Bank



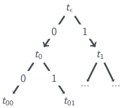
Bob | Bank



To withraw a coin, users generate their own tree

Previous Constructions

Alice

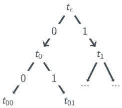


$$\xrightarrow{Acc(t_1^{(A)}, ...t_V^{(A)})}$$
$$\xleftarrow{Cert_A}$$

Bank

$Cert_A \leftarrow \mathtt{Sign}(t_i^{(A)})$

Bob



$$\xrightarrow{Acc(t_1^{(B)}, ...t_V^{(B)})}$$
$$\xleftarrow{Cert_B}$$

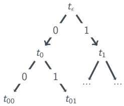
Bank

$Cert_B \leftarrow \mathtt{Sign}(t_i^{(B)})$

To withraw a coin, users generate their own tree and interact with the bank to certify them (without revealing the elements t_i)

Previous Constructions

Alice



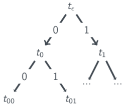
$$\xrightarrow{Acc(t_1^{(A)}, ...t_V^{(A)})}$$

$$\xleftarrow{Cert_A}$$

Bank

$Cert_A \leftarrow \texttt{Sign}(t_i^{(A)})$

Bob



$$\xrightarrow{Acc(t_1^{(B)}, ...t_V^{(B)})}$$

$$\xleftarrow{Cert_B}$$

Bank

$Cert_B \leftarrow \texttt{Sign}(t_i^{(B)})$

Users must prove that their trees are well-formed

$\Longrightarrow$ leads to complex POK during the Withraw or the Spend protocols

Our Construction

Our setting: Bilinear groups

- Bilinear groups are sets of 3 groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a map e such that

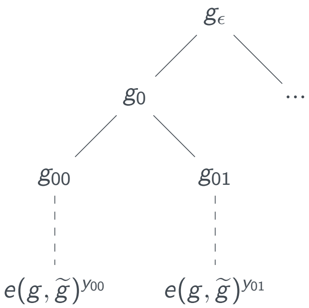
$$\forall (G_1, G_2) \in \mathbb{G}_1 \times \mathbb{G}_2 \text{ and } a, b \in \mathbb{Z}_p \; e(G_1^a, G_2^b) = e(G_1, G_2)^{a \cdot b}$$
$$e(G_1, G_2) = 1_{\mathbb{G}_T} \implies G_1 = 1_{\mathbb{G}_1} \text{ or } G_2 = 1_{\mathbb{G}_2}$$

- They play a significant role in cryptography
  - Identity Based Encryption
  - Group Signature
  - ...

- They are compatible with the Groth-Sahai proofs system

Our construction

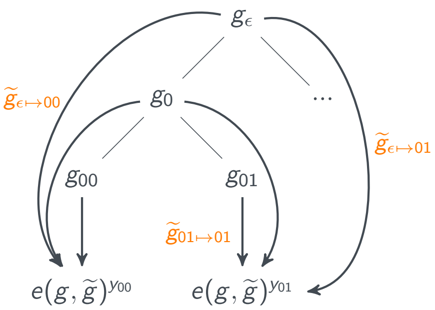
- Parameters: $g \in \mathbb{G}_1$, $\widetilde{g} \in \mathbb{G}_2$, $\forall s$, $g_s \leftarrow g^{r_s}$ for random r_s



- Our scheme makes use of only one tree, defined in the parameters
  $\Rightarrow$ No need to prove well-formedness of the tree

Our construction

- Parameters: $g \in \mathbb{G}_1$, $\widetilde{g} \in \mathbb{G}_2$, $\forall s$, $g_s \leftarrow g^{r_s}$ for random r_s



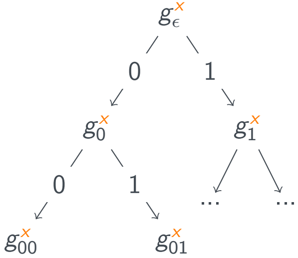
- $\forall$ s and f, $\widetilde{g}_{s \mapsto f} \leftarrow \widetilde{g}^{\frac{y_f}{r_s}}$ $\quad \Rightarrow e(g_s, \widetilde{g}_{s \mapsto f}) = e(g^{r_s}, \widetilde{g}^{\frac{y_f}{r_s}}) = e(g, \widetilde{g})^{y_f}$

Withdraw

- To withdraw a coin, users generate a secret $x \xleftarrow{\$} \mathbb{Z}_p$ and gets a certificate $Cert_x$ on it

  $\Rightarrow$ Withdrawal achievable in constant time

- Implicitly defined the users' trees as:

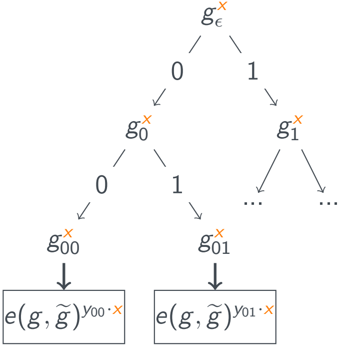


Withdraw

- To withdraw a coin, users generate a secret $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and gets a certificate $Cert_x$ on it

  $\Rightarrow$ Withdrawal achievable in constant time

- Implicitly defined the serial numbers as



Spend

To spend 2^l, the user:

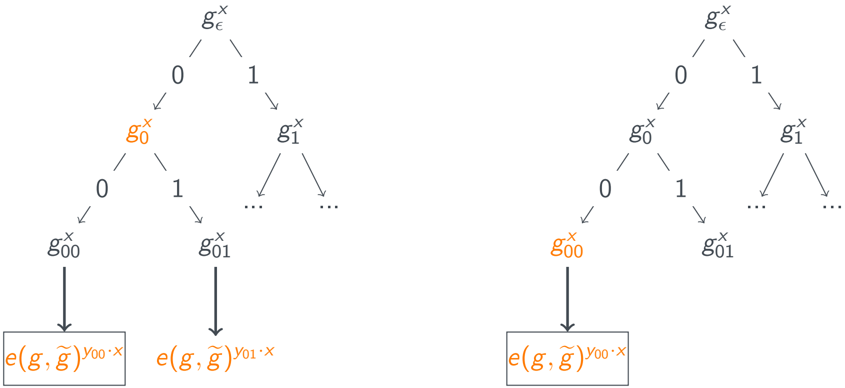
- computes:
  - $t_s \leftarrow g_s^x$
  - $\pi \leftarrow NIZK\{x, Cert_x : t_s = g_s^x \wedge Cert_x \text{ is valid}\}$
- sends (t_s, π) to the merchant who verifies π

Detection of Double-Spending

- The bank recovers the serial numbers by computing

$$e(t_s, \widetilde{g}_{s \mapsto f}) = e(g, \widetilde{g})^{y_f \cdot x}$$

- If users spend nodes 0 and 00:



Anonymity

- Transactions with the same coin involve elements $g_{s_1}^x$, $g_{s_2}^x$,...
- Linking $g_{s_i}^x$ with $g_{s_j}^x$ is hard, even with knowledge of the public parameters

$\Rightarrow$ users are unlinkable

- Our scheme can be upgraded to achieve strong unlinkability and anonymity

Achieving Anonymity

Achieving Strong Unlinkability

- The previous solution reveals the spent node s. Hiding such information rise two issues:

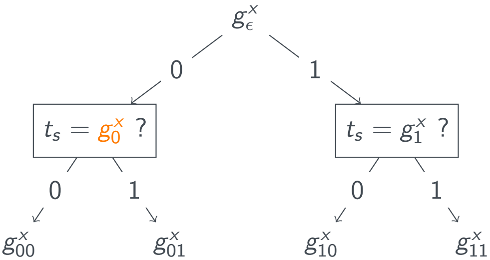
  1. Users must now prove that they use a valid g_s without revealing it
  2. The bank no longer knows which $\widetilde{g}_{s \mapsto f}$ it must use

- To fix the former, the bank will compute certificates $Cert(s)$ on every g_s

  $\Rightarrow$ allow users to prove that g_s is valid

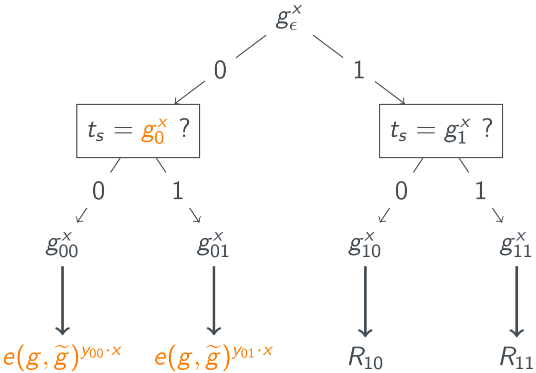
Recovering Serial Numbers

- The bank only knows the level of the spent node



Recovering Serial Numbers

- The bank only knows the level of the spent node



- It will compute $e(t_s, \widetilde{g}_{s \mapsto f})$ for every s of this level

Deposit

- The bank will recover the valid serial numbers but also invalid ones
- This increases the computational and storage cost of deposits but ensures detection of double spendings
- The resulting protocol is then strongly unlinkable and secure

Achieving Anonymity

- If a double-spending is detected the defrauder must be identified
- To achieve anonymity, this identification must be performed without a trusted entity
- We add to the previous protocol a double-spending tag which ensures the following properties:
  - Users cannot be identified as long as they are honest
  - Any defrauder can be identified by using only public information

Divisible E-Cash Made Practical?

Efficiency

In the ROM, we can achieve a remarkable efficiency:

- The data sent to the merchant consist of

Elements in	$\mathbb{Z}_p$	$\mathbb{G}_1$
	2	5

- Users can precompute most of these elements. During the transaction the user only has to perform:

Operations	$\mathbb{Z}_p$	Hash
	1	1

- We implemented this protocol on a SIM card embedded in a NFC-enabled phone. Spending values $<$ 100\$ can be performed in less than 500 ms

Efficiency

- The size of the public parameters remains reasonable: 330 KBytes for $n = 10$
- The bank must additionally store the elements $\widetilde{g}_{s \mapsto f}$ (721 KBytes)
- Our construction is the first efficient one which achieves constant time for both the withdrawal and spending protocols
- Even in the worst-case scenario of our anonymous scheme, storing the serial numbers of one million transactions requires 10 GBytes

Conclusion

Conclusion

- We proposed a practical construction for divisible E-cash
- Our construction is flexible: one can efficiently achieve different levels of anonymity
- Our scheme can be instantiated either in the ROM or in the standard model
- Our scheme is the first practical one achieving constant-time for both withdrawal and spending protocols
- Improving the efficiency of deposits of our anonymous scheme remains an open problem

Appendix

Computational Assumption

- The unlinkability of our scheme relies on the following assumption:

  Given $(g, g^{x}, g^{a}, g^{y\cdot a}, g^{z}) \in \mathbb{G}_1^5$ and $(\widetilde{g}, \widetilde{g}^{a}, \widetilde{g}^{y}) \in \mathbb{G}_2^3$, it is hard to decide whether:

  $$z = x \cdot y \cdot a \text{ or } z \text{ is random}$$

  The only way to get a product of 3 scalars is to combine $g^{y\cdot a}$ with elements of $\mathbb{G}_2$. However, x does not appear in the latter.

Double-Spending Tag

- Each node s is now associated with a pair $(g_s, h_s) \leftarrow (g^{r_s}, h^{r_s})$ for some $h \in \mathbb{G}_1$
- To spend 2^l, the user whose public key is $\texttt{upk} \in \mathbb{G}_1$ also computes:

$$v_s \leftarrow \texttt{upk} \cdot h_s^x$$

and proves its validity

Identification

- A double-spending involves two nodes s and s' with a common leaf f. Therefore, we have:

$$e(h_s, \widetilde{g}_{s \mapsto f}) = e(h_{s'}, \widetilde{g}_{s' \mapsto f})$$

- Then, $e(v_s, \widetilde{g}_{s \mapsto f}) \cdot e(v_{s'}, \widetilde{g}_{s' \mapsto f})^{-1} = e(\mathtt{upk}, \widetilde{g}_{s \mapsto f} \cdot \widetilde{g}_{s' \mapsto f}^{-1})$

- The defrauders can thus be identified by exhaustive search