Proxy Re-Signature in the Standard Model

Sherman Chow  Raphael Phan

Presented by Xuhua Ding

New York University

ISC 2008 Sep 18 2008

Loughborough University
Rundown

• Introduction and applications
• Properties of Proxy Re-Signature (PRS)
• Security notions of different PRS schemes
• Analysis of a Bidirectional PRS scheme
• Homomorphic Compartment Signature (HCS)
• PRS from HCS
• Forward-Security and Temporary Delegation
• Proxy re-cryptography
  – delegating transformation right of cryptographic objects to a *semi-trusted proxy*
  – a cryptographic task which can only be completed by a delegator now becomes a task that can be completed by a delegatee
Whatever you sign, I’ll sign

• Proxy re-signature
  – signatures signed by a delegator can be transformed (by a proxy) into ones signed by a delegatee
  – without allowing the proxy to sign on any other messages
  – without further involvement of the delegator/delegatee
Some Applications

- Public key certificate management
  - transform a certificate signed by some key to one that can be verified by a public key which is already “trusted” by the verifier
  - cross-certification
- A kind of “group signature”
  - transform a signature signed by an employee to one that can be verified by the corporate key
KeyGen, Sign, Verify: like a standard signature

ReKey
- takes (an optional) delegatee’s private key \( sk_A \)
- a delegator’s private key \( sk_B \)
- the corresponding public key \((pk_A, pk_B)\)
- outputs \(rk_{A->B}\)
- which transforms A’s signature into B’s
- remember that B is the delegator, since \(rk_{A->B}\) will be used by ReSign to produce B’s signature
KeyGen, Sign, Verify, ReKey

ReSign
- takes the transformation key $rk_{A\rightarrow B}$
- a transformable signature $\sigma_A$
- a public key $pk_A$
- a message $m$
- outputs $\sigma_B$
Private Proxy, Non-interactive

- **Private Proxy**
  - The transformation key $rk_{A->B}$ can be kept private even seeing many pair of signature and its transformation
  - (public proxy):
    - e.g. a signature $\sigma_A$ and its transformation $\sigma_B$ gives $rk_{A->B}$
    - Everyone is a proxy, not desirable

- **Non-interactive**
  - Transformation key can be created w/o A’s secret key
  - (interactive): requires A’s help
Unidirectional, Multi-use

- Unidirectional
  - having $rk_{A\rightarrow B}$ does not help the proxy to get $rk_{B\rightarrow A}$
  - (bidirecitonal):
    - must be interactive
    - easier to design (a scheme will be analyzed later)

- Multi-use
  - Signature output by ReSign is still transformable
  - (single-use): ReSign must take a signature output by Sign, but not by ReSign
Transparent, Non-transitive, ...

- Transparent
  - Signatures generated by Sign and those generated by ReSign are computationally indistinguishable

- Non-transitive
  - The proxy alone cannot re-delegate signing right, e.g. $rk_{A\rightarrow B} + rk_{B\rightarrow C}$ would not give $rk_{A\rightarrow C}$

- Temporary
  - $rk_{A\rightarrow B}$ can be expired
Security for Different PRS Schemes

- 6 major properties, 2 possibilities each
  - 64 different schemes
  - Which notion must be considered in which case?
  - Which notion may not make sense in some cases?
- PRS w/ external security ensures private-proxy
- Delegator security is for a non-transparent PRS
- Bidirectional limited-proxy security is proposed
- Refers to the paper for details
Waters Signature

- Define $F(m_1 m_2 \ldots m_n) = u' u_1^{m_1} u_2^{m_2} \ldots u_n^{m_n}$
- $e()$ is a bilinear map, $e(g^a, g^b) = e(g, g)^{ab}$
- KeyGen: pick $sk \in \mathbb{Z}_p$, $pk = e(g, g)^{sk}$
- Sign: pick $r \in \mathbb{Z}_p$, output $(\sigma_1 = g^{sk} F(m)^r, \sigma_2 = g^r)$
- Verify: $e(\sigma_1, g) = pk \ e(F(m), \sigma_2)$
Review of A Bi-directional PRS

- Proposed by Shao et al. in Indocrypt ‘07
- Underlying signature: \( (\sigma_1 = g^{sk} F(m)^r, \sigma_2 = g^r) \)
- ReKey is just \( rk_{A \rightarrow B} = sk_B / sk_A \)
  - Its inverse is computable (prime order group)
- ReSign just exponentiates the signature with the transformation key \( rk \), i.e. \( (\sigma_1^{rk}, \sigma_2^{rk}) \)
Design (Flaw)

• One cannot tell the $rk$ exponent from $\sigma$ and $\sigma^{rk}$
  – Their ReSign algorithm does not bother to introduce its own randomness
• Security of Waters signature relies on random $r$
• Insecurity originated from the deterministic ReSign algorithm, i.e. $r$ in the exponent is fixed
  – Transforming a signature using random factor $r$ always result one implicitly using $r \times sk_B / sk_A$
Our First Attack

• Find 4 messages with hash values such that
  – $h_4 = h_1 + h_2 - h_3$ but $h_4 \neq h_1$
  – These messages can be found with high probability

• Sign $h_1, h_2, h_3$, all uses the same randomness $r$

• Ask ReSign oracle to transform into sig. by $sk'$

• Recall that $\sigma_1 = g^{sk'}F(m)r'$, and all transformed signatures are using the same random factor

• $g^{sk'} F(h_1)r' \times g^{sk'} F(h_2)r' / g^{sk'} F(h_3)r' = g^{sk'} F(h_4)r'$
Key Recovery Attack

• The above attack manipulates the bit strings s.t. an addition and then a subtraction gives a valid bit string different from the original one.
• Taking a step further, we can cancel the whole message part and get back the private key!
• Details in the paper, we remark that Kim-Yie-Lim (Intl. J. of Net. Sec.) found the same attack
• Possible fix: re-randomization in ReSign
How to design a unidirectional PRS

- Previous random-oracle based scheme by Ateniese and Hohenberger
- Map signatures from $\mathbb{Z}_p$ to $G$ equipped with bilinear maps
  - One cannot invert pairing, which intuitively matches the idea of unidirectional transformation
- Inherently requires proof-of-knowledge
Hierarchical Signature

• The message to be signed consists of \( h \) blocks
• A signature on \((m_1; m_2; \ldots; m_i)\) can act as a restricted private key that enables the signing of any extension \((m_1; m_2; \ldots; m_j)\)
• E.g. the secret key gives a signature on \(m_1\), this signature can be used (without using the secret key) to sign on \(m_1; m_2\)
• Generalize hierarchical signature
• Take $h = 2$, and let * denote a “null message”
• The secret key can sign on either
  – $(m_1; *)$
  – $(*; m_2)$ or
  – $(m_1; m_2)$ for any $m_1, m_2$
• It is also possible to sign on $(m_1; m_2)$ by either
  – $(m_1; *)$ or
  – $(*; m_2)$
A New Approach for Unidirectional PRS

- To make unidirectional possible, we sign on the delegation relationship $A\rightarrow B$, then we use this as a transformation key.
- "Compartment" property gives us the flexibility to sign on the message in one level, or this delegation relationship in another level.
The Missing Pieces

- How to cancel the secret key like \( rk_{A\rightarrow B} = sk_B/sk_A \) in bidirectional PRS schemes?
  - Homomorphically

\( HSign_{sk}(m_1, m_2) \cdot HSign_{sk'}(m_1, m_2) = HSign_{sk+sk'}(m_1, m_2) \)

- How to ensure private-proxy property?
  - Re-randomization

\( HSign_{sk}(m_1, m_2; r_1, r_2) \cdot HSign_{sk'}(m_1, m_2; r'_1, r'_2) = HSign_{sk+sk'}(m_1, m_2; r_1 + r'_1, r_2 + r_2) \)
While homomorphic compartment signature (HCS) is more general, many hierarchical signature schemes are actually HCS schemes.

These schemes use different parameters for different levels, and uses only commutative multiplications to “glue” up different components.

E.g. signature schemes of Boneh-Boyen-Goh, and Boyen-Shacham-Shen-Waters
System parameters:
- \( F(m_1m_2...m_n) = u'^1u_1^{m_1}u_2^{m_2}...u_n^{m_n} \)
- \( F'(m_1m_2...m_n) = v'^1v_1^{m_1}v_2^{m_2}...v_n^{m_n} \)

KeyGen: pick \( sk \in \mathbb{Z}_p \), \( pk = e(g, g)^{sk} \)

Sign:
- pick \( r, t \in \mathbb{Z}_p \)
- output \( (\sigma_1 = g^{sk}F(m)^r F'(m')^t , \sigma_2 = g^r , \sigma_3 = g^t) \)

Verify: \( e(\sigma_1, g) = pk e(F(m), \sigma_2) e(F'(m'), \sigma_3) \)
Generic Proxy Re-Signature

- Let CS denote the compartment signature
- Sign: $\text{CS.HSign}_{sk}(*, m; 0, r)$
- ReKey (Interactive):
  - Compute $m_{A\rightarrow B} = H(pk_B)$
  - Pick $r_A$, compute $\sigma_A = \text{CS.HSign}_{skA}(m_{A\rightarrow B}, *; r_A, 0)$
  - Pick $r_B$, compute $\sigma_B = \text{CS.HSign}_{skB}(m_{A\rightarrow B}, *; r_B, 0)$
  - Transformation key is $\sigma_B / \sigma_A$
  - $rk_{A\rightarrow B} = \text{CS.HSign}_{skB-skA}(m_{A\rightarrow B}, *; r_B - r_B, 0)$
Generic Proxy Re-Signature (cont.)

- Sign: $\sigma = \text{CS.HSign}_{sk_A}(\ast, m; 0, r)$
- ReKey: $rk_{A\rightarrow B} = \text{CS.HSign}_{sk_B-sk_A}(m_{A\rightarrow B}, \ast; r_B - r_B, 0)$
- ReSign:
  - Pick $r'$, output $\sigma * rk_{A\rightarrow B} * \text{CS.HSign}_0(\ast, m; 0, r')$
- Verify:
  - Transformable signature: $\text{CS.Verify}(\sigma, (\ast, m))$
  - Transformed signature: $\text{CS.Verify}(\sigma, (H(pk), m))$
Hierarchical signature can be used to build forward-secure signature scheme.

Generalizing the proposed generic construction, we get forward-secure PRS.
A New Approach for Temporary Delegation

- The proposed construction certifies the delegation relationship by signature
- The delegation thus can be made time-limited
  - the transformation key created at time $t$ can only enable the transformation of signatures for time $t$, but not any signature issued at any other time
- Our solution just requires delegating a transformation key for the new time period to the proxy
  - instead of assuming all users can access an authenticated copy of the new global parameter [Ateniese-Hohenberger]
Thank you very much!

- Questions and comments are welcome.
- schow@cs.nyu.edu
Security Model

• Proposed by Ateniese and Hohenberger
• Recall that the system has 3 kinds of entities
  – Proxy
  – Delegation partners
    • Delegator
    • Delegatee
• External security
  – attack launched from parties outside the system
  – i.e. neither the delegation partners nor the proxy
• For a public-proxy PRS, no one is outsider!
• PRS with external security ensures private-proxy
Insider Security

• Limited-Proxy Security
  – Proxy is the adversary, but its power is limited to itself, i.e. not colluding with any delegation partner

• Delegatee Security
  – Collusion of delegators and proxy
  – Delegatee is to be attacked

• Delegator Security
  – Collusion of delegates and proxy
  – Delegator is to be attacked
Suppose user to be attacked has key \((pk', sk')\)

OSign, OReKey denotes the signing oracle and the transformation key oracle

\(pk\) usually refers to an arbitrary public key
Limited Proxy Security

- Only see the public keys of all users
- Signing oracle accesses of all users
- ReKey oracle accesses (proxy is the adversary)
- Winning condition:
  - Forgery does not come from OSign of $pk'$
  - Forgery does not come from OSign of $pk$, if OReKey($pk$, $pk'$) has been queried for any $pk$
- Note that the adversary can get $rk_{pk->pk'}$
• Has the private keys of all but 1 users
• Signing oracle of $pk'$ (others are trivial)
• ReKey oracle for $(pk_0, pk)$ as long as $pk \neq pk'$
  – Otherwise $pk$ becomes a delegator
  – Recall a limited-proxy adversary *can* get $rk_{pk \rightarrow pk'}$
• Winning condition (usual):
  – Forgery does not come from OSign of $pk'$
Delegator Security

- Has the private keys of all but 1 users
- Signing oracle of $pk'$ (others are trivial)
- ReKey oracle queries for all kind
  - Different from delegatee security
- Winning condition (usual):
  - Forgery does not come from OSign of $pk'$
  - Forgery is a “untransformed” signature
- The notion for a non-transparent scheme