Universal Re-Encryption of Signatures

Marek Klonowski, Mirosław Kutyłowski, Anna Lauks, Filip Zagórski

Wrocław University of Technology,
Mikulašska Kryptobesidka 2004

Goals

- anonymous communication
- but some control allowing inappropriate data to be discarded,
  without violating anonymity and secrecy

Assumptions

Each message *m* entering the system

- is encrypted with the public key of the recipient

Assumptions

Each message m entering the system

- is encrypted with the public key of the recipient
- must be admitted by an Authority, which signs a ciphertext of m provided that
  - it knows m (and regards m as legal), or
  - obtains m from a trusted user

Assumptions

Anonymity: a ciphertext of m + a signature of the ciphertext issued by the Authority **are recoded** by each server on their route.

Assumptions

Anonymity a ciphertext of m + a signature of the ciphertext issued by the Authority **are recoded** by each server on their route.

Control signature's validity can be checked at any time, regardless of recoding.

Unknown:

- the plaintext message m
- the destination

Example applications

- control of information flow

Example applications

- control of information flow
- against spam in anonymous email systems- the system would refuse to process emails that have not been subject to a priori control

Example applications

- control of information flow
- against spam in anonymous email systems- the system would refuse to process emails that have not been subject to a priori control
- against malicious mix servers in network of mixes

Example applications

- control of information flow
- against spam in anonymous email systems- the system would refuse to process emails that have not been subject to a priori control
- against malicious mix servers in network of mixes
- to confirm that a given message was processed by a particular anonymizer

Scenarios

- A server can check signature's validity with the public key of the Authority (**RSA-URE Signature**)

Scenarios

- A server can check signature's validity with the public key of the Authority (**RSA-URE Signature**)
- Each server can check signature's validity only in cooperation with the Authority (**Undeniable URE Signature**). Possible outcomes:
  - valid signature
  - invalid signature
  - the Authority is cheating

URE - Universal Re-Encryption

P. Golle, M. Jakobsson, A. Juels, P. Syverson

Assumptions like for ElGamal encryption:

- p is a prime number with hard discrete logarithm problem
- g - generator of $\mathbb{Z}_p^*$,
- $x < p - 1$ - private key of a recipient
- $y = g^x \mod p$ - public key of the recipient

Universal Re-Encryption

Encryption: (all operations modulo *p*)

k_0, k_1 - random

A ciphertext of *m*:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1) := \left(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}\right)$$

Universal Re-Encryption

Encryption: (all operations modulo p)
k_0, k_1 - random
A ciphertext of m:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1) := \left(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}\right)$$

Re-encryption:
k'_0, k'_1 - random
The message after re-encryption:

$$\left(\alpha_0 \cdot \alpha_1^{k'_0}, \beta_0 \cdot \beta_1^{k'_0}; \alpha_1^{k'_1}, \beta_1^{k'_1}\right)$$

Decryption

$(\alpha_0, \beta_0; \alpha_1, \beta_1)$

The recipient computes:

$$m_0 = \frac{\alpha_0}{\beta_0^x}$$

$$m_1 = \frac{\alpha_1}{\beta_1^x}$$

A message $m = m_0$ is accepted $\Leftrightarrow m_1 = 1$

RSA-URE signature

An extension of a construction from Golle's "Reputable Mix Networks"

- $N = pq$ is an RSA number, g - a generator of $\mathbb{Z}_N^*$,
- key pair of the Authority: e, d (private), where $e \cdot d = 1 \bmod \phi(N)$
- recipient's keys: x - private key, $y = g^x$ - public key
- additional parameters:
  $\hat{g} = g^d$,
  $\hat{y} = y^d = g^{dx} = \hat{g}^x$

Signature creation

k_0, k_1 - random

Signature creation

k_0, k_1 - random

The signed ciphertext:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= (m \cdot y^{k_0}, g^{k_0};$$

Signature creation

k_0, k_1 - random

The signed ciphertext:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1};$$

Signature creation

k_0, k_1 - random

The signed ciphertext:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; m^d \cdot \hat{y}^{k_0}, \hat{g}^{k_0};$$

Signature creation

k_0, k_1 - random

The signed ciphertext:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= \left(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; m^d \cdot \hat{y}^{k_0}, \hat{g}^{k_0}; \hat{y}^{k_1}, \hat{g}^{k_1}\right)$$

Re-encryption

signed message: $(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3)$

Re-encryption

signed message: $(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3)$

k_0', k_1' - random

after re-encryption:

$$\Big(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'};$$

Re-encryption

signed message: $(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3)$

k'_0, k'_1 - random

after re-encryption:

$$\left(\alpha_0 \cdot \alpha_1^{k'_0}, \beta_0 \cdot \beta_1^{k'_0}; \alpha_1^{k'_1}, \beta_1^{k'_1};\right.$$

Re-encryption

signed message: $(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3)$

k'_0, k'_1 - random

after re-encryption:

$$\left(\alpha_0 \cdot \alpha_1^{k'_0}, \beta_0 \cdot \beta_1^{k'_0}; \alpha_1^{k'_1}, \beta_1^{k'_1}; \alpha_2 \cdot \alpha_3^{k'_0}, \beta_2 \cdot \beta_3^{k'_0};\right.$$

Re-encryption

signed message: $(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3)$

k_0', k_1' - random

after re-encryption:

$$\left(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'}; \alpha_1^{k_1'}, \beta_1^{k_1'}; \alpha_2 \cdot \alpha_3^{k_0'}, \beta_2 \cdot \beta_3^{k_0'}; \alpha_3^{k_1'}, \beta_3^{k_1'}\right)$$

First re-encryption

a server gets:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) =$$
$$= (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; m^d \cdot \hat{y}^{k_0}, \hat{g}^{k_0}; \hat{y}^{k_1}, \hat{g}^{k_1})$$

First re-encryption

a server gets:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) =$$
$$= (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; m^d \cdot \hat{y}^{k_0}, \hat{g}^{k_0}; \hat{y}^{k_1}, \hat{g}^{k_1})$$

and computes:

$$(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'}; \alpha_1^{k_1'}, \beta_1^{k_1'}; \alpha_2 \cdot \alpha_3^{k_0'}, \beta_2 \cdot \beta_3^{k_0'}; \alpha_3^{k_1'}, \beta_3^{k_1'}) =$$

First re-encryption

a server gets:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) =$$
$$= (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; m^d \cdot \hat{y}^{k_0}, \hat{g}^{k_0}; \hat{y}^{k_1}, \hat{g}^{k_1})$$

and computes:

$$(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'}; \alpha_1^{k_1'}, \beta_1^{k_1'}; \alpha_2 \cdot \alpha_3^{k_0'}, \beta_2 \cdot \beta_3^{k_0'}; \alpha_3^{k_1'}, \beta_3^{k_1'}) =$$
$$= (m \cdot y^{k_0} \cdot (y^{k_1})^{k_0'}, \ \ldots\ ; \ \ldots\ , \ \ldots\ ;$$
$$m^d \cdot \hat{y}^{k_0} \cdot (\hat{y}^{k_1})^{k_0'}, \ \ldots\ ; \ \ldots\ , \ \ldots\ ) =$$

First re-encryption

a server gets:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) =$$
$$= (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; m^d \cdot \hat{y}^{k_0}, \hat{g}^{k_0}; \hat{y}^{k_1}, \hat{g}^{k_1})$$

and computes:

$$\begin{aligned}
&(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'}; \alpha_1^{k_1'}, \beta_1^{k_1'}; \alpha_2 \cdot \alpha_3^{k_0'}, \beta_2 \cdot \beta_3^{k_0'}; \alpha_3^{k_1'}, \beta_3^{k_1'}) = \\
&= (m \cdot y^{k_0} \cdot (y^{k_1})^{k_0'}, \ \ldots \ ; \ \ldots \ , \ \ldots \ ; \\
&\quad m^d \cdot \hat{y}^{k_0} \cdot (\hat{y}^{k_1})^{k_0'}, \ \ldots \ ; \ \ldots \ , \ \ldots \ ) = \\
&= (m \cdot y^{k_0 + k_1 \cdot k_0'}, \ \ldots \ ; \ \ldots \ , \ \ldots \ ; \\
&\quad m^d \cdot \hat{y}^{k_0 + k_1 \cdot k_0'}, \ \ldots \ ; \ \ldots \ , \ \ldots \ )
\end{aligned}$$

Verification of RSA-URE signature

If a RSA-URE signature is correct, then for some k we have:

$$\alpha_0 = m \cdot y^k \quad \text{and} \quad \alpha_2 = m^d \cdot \hat{y}^k$$

so $\alpha_2 = \alpha_0^d$.

Verification of RSA-URE signature

If a RSA-URE signature is correct, then for some k we have:

$$\alpha_0 = m \cdot y^k \quad \text{and} \quad \alpha_2 = m^d \cdot \hat{y}^k$$

so $\alpha_2 = \alpha_0^d$.

Hence the verifier accepts the signature iff $\alpha_0 = \alpha_2^e$

Blind creation of RSA-URE signature

Alice chooses a random k, $\mathrm{GCD}(k, N) = 1$,

$$t := \left(m \cdot y^{k_0} \cdot k^e, g^{k_0} \cdot k^e; y^{k_1} \cdot k^e, g^{k_1} \cdot k^e\right)$$

Blind creation of RSA-URE signature

Alice chooses a random k, $\mathrm{GCD}(k, N) = 1$,

$$t := \left(m \cdot y^{k_0} \cdot k^e, g^{k_0} \cdot k^e; y^{k_1} \cdot k^e, g^{k_1} \cdot k^e\right)$$

and sends t to the Authority. The Authority raises each component to the power d:

$$t^d = \left(m^d \cdot y^{d \cdot k_0} \cdot k, g^{d \cdot k_0} \cdot k; y^{d \cdot k_1} \cdot k, g^{d \cdot k_1} \cdot k\right)$$

and sends to Alice.

Blind creation of RSA-URE signature

Alice chooses a random k, $\mathrm{GCD}(k,N)=1$,

$$t := \left(m \cdot y^{k_0} \cdot k^e, g^{k_0} \cdot k^e; y^{k_1} \cdot k^e, g^{k_1} \cdot k^e\right)$$

and sends t to the Authority. The Authority raises each component to the power d:

$$t^d = \left(m^d \cdot y^{d \cdot k_0} \cdot k, g^{d \cdot k_0} \cdot k; y^{d \cdot k_1} \cdot k, g^{d \cdot k_1} \cdot k\right)$$

and sends to Alice.

Alice removes k

$$\frac{t^d}{k} = \left(m^d \cdot y^{d \cdot k_0}, g^{d \cdot k_0}; y^{d \cdot k_1}, g^{d \cdot k_1}\right)$$

Blind creation of RSA-URE signature

Alice chooses a random k, $\mathrm{GCD}(k,N) = 1$,

$$t := \left(m \cdot y^{k_0} \cdot k^e, g^{k_0} \cdot k^e; y^{k_1} \cdot k^e, g^{k_1} \cdot k^e\right)$$

and sends t to the Authority. The Authority raises each component to the power d:

$$t^d = \left(m^d \cdot y^{d \cdot k_0} \cdot k, g^{d \cdot k_0} \cdot k; y^{d \cdot k_1} \cdot k, g^{d \cdot k_1} \cdot k\right)$$

and sends to Alice.

Alice removes k

$$\frac{t^d}{k} = \left(m^d \cdot y^{d \cdot k_0}, g^{d \cdot k_0}; y^{d \cdot k_1}, g^{d \cdot k_1}\right)$$

Alice creates a standard RSA-URE signature:

$$\left(my^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; m^d y^{dk_0}, g^{dk_0}; y^{dk_1}, g^{dk_1}\right)$$

Undeniable URE-signatures

based on scheme of D. Chaum, H. van Antwerpen

Undeniable URE-signatures

based on scheme of D. Chaum, H. van Antwerpen

Notation, Assumptions:

- p, q - primes, $p = 2q + 1$
- g - generator of G - a subgroup of $\mathbb{Z}_p^*$ of order q
- $d \in \{1, \ldots, q-1\}$ - random signing key
- e - public key, $e := g^d \bmod p$

Undeniable URE-signatures

based on scheme of D. Chaum, H. van Antwerpen

Notation, Assumptions:

- p, q - primes, $p = 2q + 1$
- g - generator of G - a subgroup of $\mathbb{Z}_p^*$ of order q
- $d \in \{1, \ldots, q-1\}$ - random signing key
- e - public key, $e := g^d \mod p$

Signature under message m:

$$s := m^d \mod p$$

Signature verification

- Verifier chooses $i, j \in \{1, \ldots, q-1\}$ at random, computes: $z := s^i e^j \mod p$ and presents z to Alice

Signature verification

- Verifier chooses $i, j \in \{1, \ldots, q-1\}$ at random, computes: $z := s^i e^j \mod p$ and presents z to Alice
- Alice computes $w := (z)^{d^{-1} \mod q} \mod p$ and presents w to Verifier

Signature verification

- Verifier chooses $i, j \in \{1, \ldots, q-1\}$ at random, computes: $z := s^i e^j \mod p$ and presents z to Alice
- Alice computes $w := (z)^{d^{-1} \bmod q} \mod p$ and presents w to Verifier
- Verifier computes $w' := m^i g^j \mod p$ and accepts the signature s iff $w = w'$

... further procedures ...

Undeniable URE-signatures

Notation, Assumptions:

- m - message
- p, q, g, G - chosen as in the previous case
- $d \in \{1, \ldots, q-1\}$ - private signing key of the Authority
- $e = g^d \bmod p$ - public key of the Authority
- x - private key of a recipient
- $y = g^x$ - public key of the recipient

URE-signature creation

(a version of $s := m^d$)

URE-signature creation

(a version of $s := m^d$)

k_0, k_1 - random

URE-signature creation

(a version of $s := m^d$)

k_0, k_1 - random

a signed ciphertext of m:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= \big(m \cdot y^{k_0}, g^{k_0};$$

URE-signature creation

(a version of $s := m^d$)

k_0, k_1 - random

a signed ciphertext of m:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= \big(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1};$$

URE-signature creation

(a version of $s := m^d$)

k_0, k_1 - random

a signed ciphertext of m:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; s \cdot y^{dk_0}, g^{dk_0};$$

URE-signature creation

(a version of $s := m^d$)

k_0, k_1 - random

a signed ciphertext of m:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1; \alpha_2, \beta_2; \alpha_3, \beta_3) :=$$

$$= \left(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}; s \cdot y^{dk_0}, g^{dk_0}; y^{dk_1}, g^{dk_1}\right)$$

Blind creation of s

Alice chooses values k_0 and k_1, computes

$$\left(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}\right)$$

and presents it to the Authority.

Blind creation of s

Alice chooses values k_0 and k_1, computes

$$\left(m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1}\right)$$

and presents it to the Authority.

The Authority computes the missing components by rising each number to the power d.

Why verification procedures can be implemented?

- the original verification procedure checks that the discrete logarithms of y and s with respect to g and m are the same

Why verification procedures can be implemented?

- the original verification procedure checks that the discrete logarithms of y and s with respect to g and m are the same
- our encoding is valid if discrete logarithms are the same for 4 pairs of components

Why verification procedures can be implemented?

- the original verification procedure checks that the discrete logarithms of y and s with respect to g and m are the same
- our encoding is valid if discrete logarithms are the same for 4 pairs of components
- re-codings preserve equality of discrete logarithms

Conclusions and open problems

- Constructing URE signatures is relatively straightforward for signature schemes that are based on exponentiations only.

Conclusions and open problems

- Constructing URE signatures is relatively straightforward for signature schemes that are based on exponentiations only.
- How to universally re-encrypt signatures such as DSA?

Thank you for attention!