Blind signature as a shield against backdoors in smart-cards

Liliya Akhmetzyanova, Evgeny Alekseev, Alexandra Babueva,
Andrey Bozhko, Stanislav Smyshlyaev

CRYPTOPRO

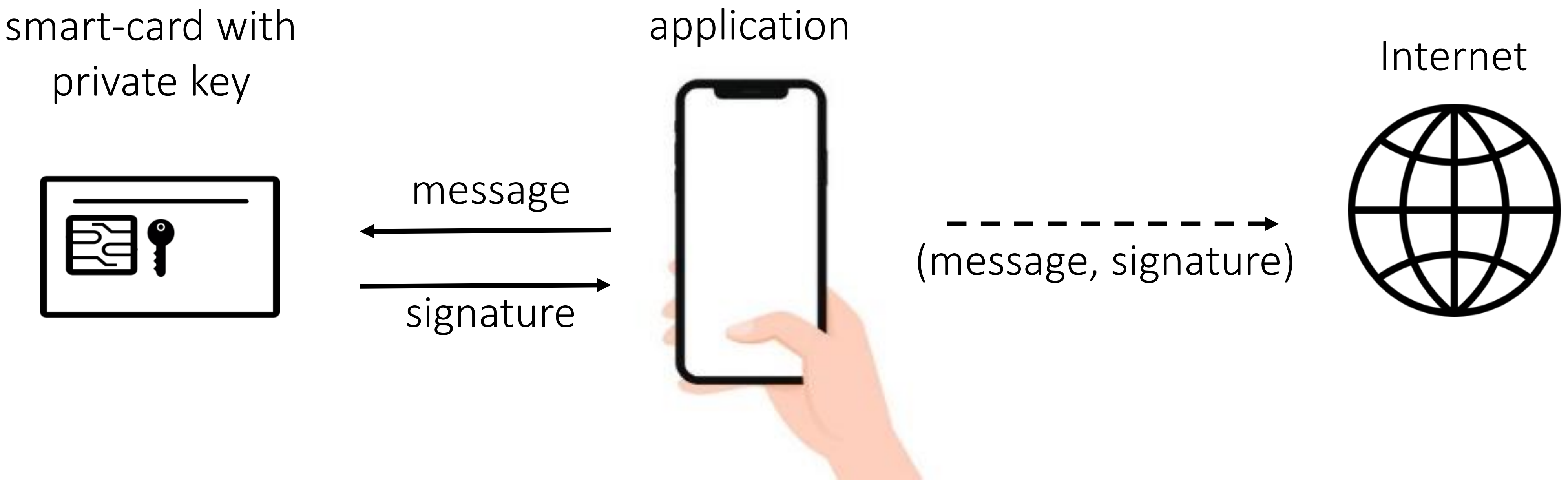
2023 CTCRYPT

Outline

1. Motivation
2. Related work
3. Blind signatures
4. Our contribution
5. Open problems

Outline

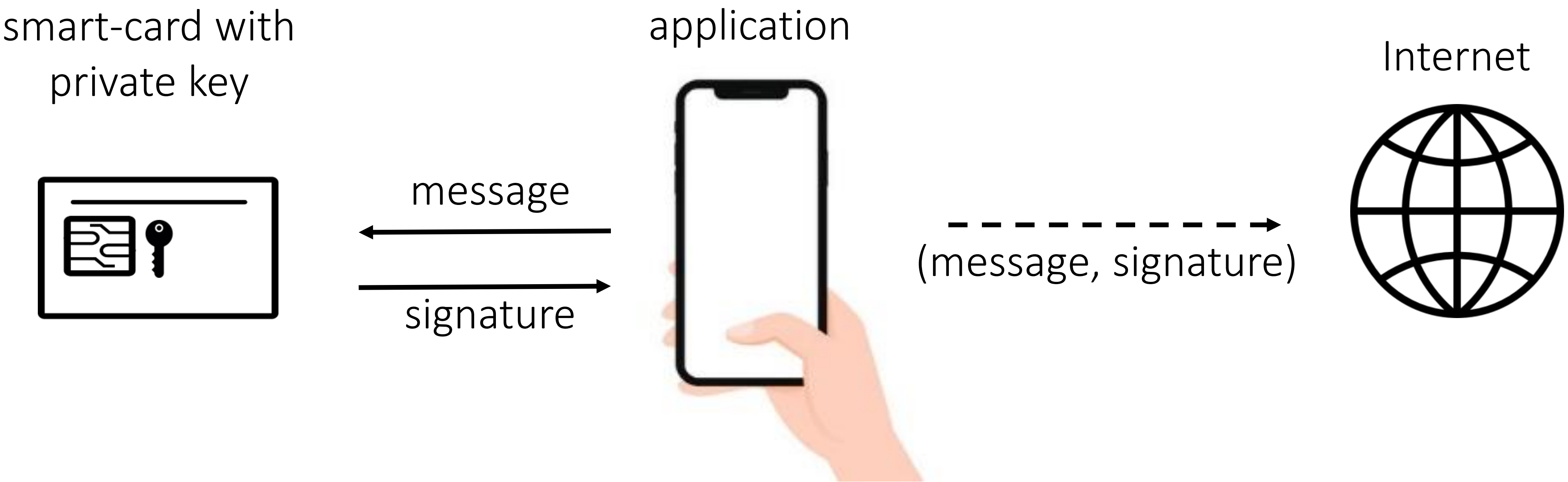
Smart-cards: secure key management



The uploaded user signing key is protected against adversary-thief which can get physical access to smart-card:

- Engineering protection against physical key extraction;
- Password-based protected access to signing API (e.g. PAKE).

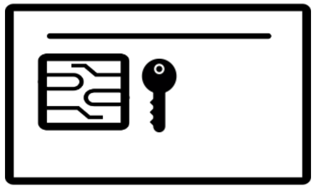
Smart-cards : security issues



The signing code is often hardwired into smart-card microchips and cannot be openly verified. This makes it possible for unscrupulous developers to implement a malicious code.

Smart-cards : security issues

smart-card with
private key



$k \xleftarrow{R} D$

$r = kP.x$

$s = ke + dr$

e

(s, r)

application

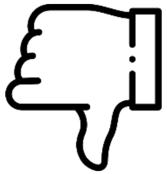
$e, (s, r)$

Internet

Try each $k \in D$:

$d = r^{-1}(s - ke)$

until $dP = Q$



Example: in case of using the GOST signature scheme, malicious smart-card can sample low-entropy one-time values k allowing the developer to recover the user key from one correct signature.

Outline

CRYPTOPRO

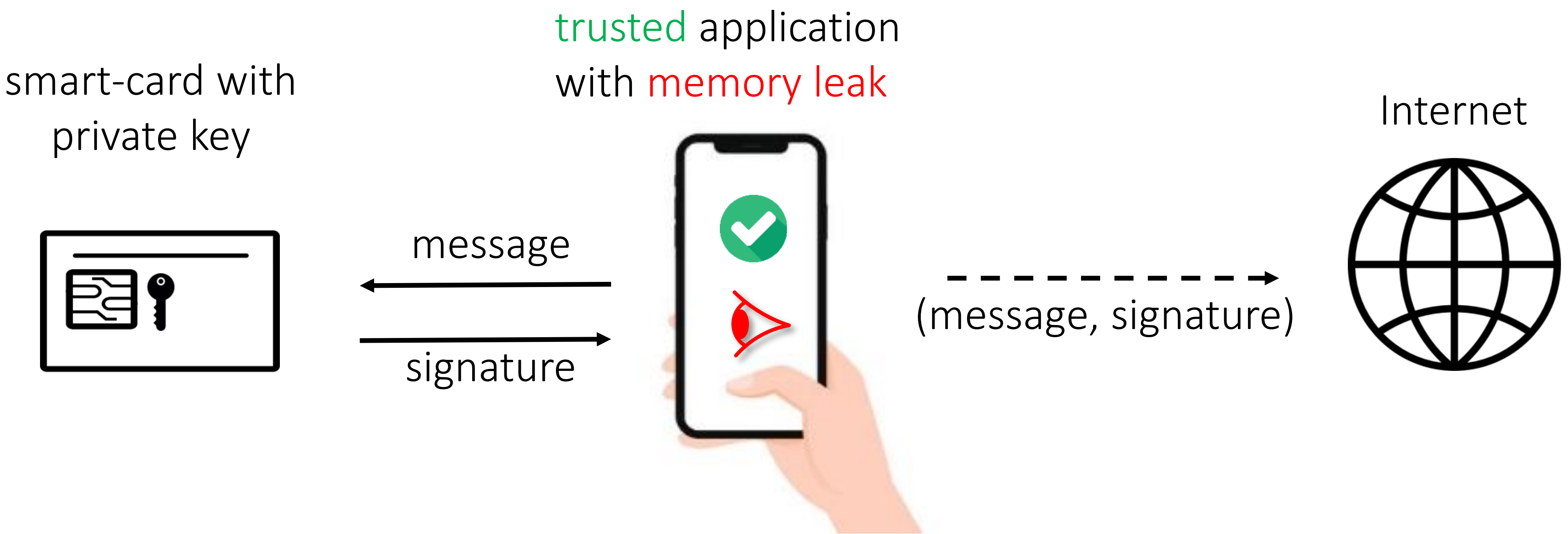
Previously on …

Akhmetzyanova L., Alekseev E., Bozhko A., Smyshlyaev S., "**Secure implementation of digital signature using semi-trusted computational core**", Mathematical Aspects of Cryptography, 2021.

It proposes:

1. to consider two type of adversaries:
    - External (to smart-card) adversary
    - Adversary with agent (malicious smart-card)
2. solution for the GOST signature scheme to protect against such adversaries.

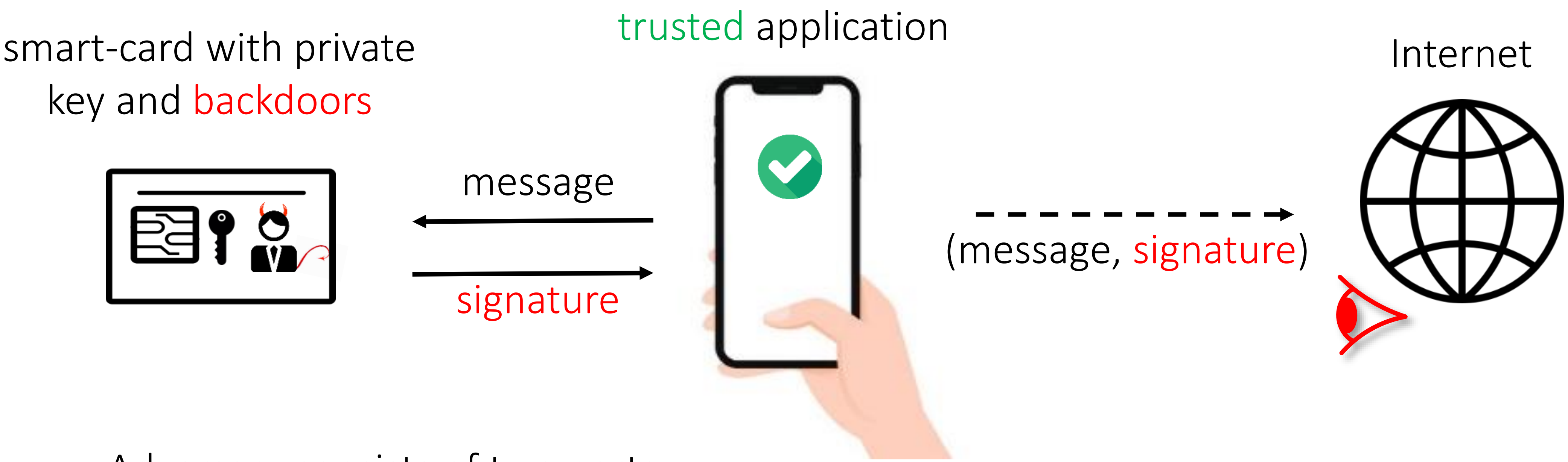
External adversaries



Honest-but-curious adversary acting on the application side (computer virus).

Goal: to make a forgery, in particular, by key recovery

Adversary with agent



Adversary consists of two parts:

1. an active adversary on the smart-card side (smart-card with backdoor).
2. a passive adversary collecting correct signatures (backdoor implementer).

Goal: to make a forgery, in particular, by key recovery

Solution

Idea of the solution: additional usage of Schnorr ZKP, the smart-card proves to the application that it used the «correct» high-entropy one-time value without revealing it.

But it has two drawbacks:

- allows to protect against the *semi-trusted* smart-card only;
- not secure if the smart-card can terminate the signing process with the error (attack was proposed).

$\mathrm{uSign}[\mathrm{Token}, \mathrm{CSP}](m, (d, Q)) \to (k, (r, s))$		
Token : d	secure channel	CSP : Q, m
$seed_1 \xleftarrow{\mathcal{U}} \{1, \ldots q-1\}$		
		$seed_2 \xleftarrow{\mathcal{U}} \{1, \ldots q-1\}$
		$h \leftarrow \mathrm{H}(m)$
	$\xleftarrow{seed_2, h}$	
$k = (seed_1 + seed_2) \bmod q$		
if $k = 0$ **return** $\perp$		
$e = h \bmod q$		
if $e = 0$ **then** $e = 1$		
$C = kP$		
$r = x_C \bmod q$		
$s = (rd + ke) \bmod q$		
$u \xleftarrow{\mathcal{U}} \{1, \ldots q-1\}$		
$W = uP$		
$w = x_W \bmod q$		
	$\xrightarrow{(r, s), w}$	
		$e = h \bmod q$
		if $e = 0$ **then** $e = 1$
		$C = e^{-1}(sP - rQ)$
		if $x_C \neq r \bmod q$ **return** $\perp$
		$l \xleftarrow{\mathcal{U}} \{1, \ldots q-1\}$
	$\xleftarrow{l}$	
$v = (u + lk) \bmod q$		
	$\xrightarrow{v}$	
		$W = vP - lC$
		if $x_W \neq w \bmod q$ **return** $\perp$
return k		**return** (r, s)

Solution

Idea of the attack: smart-card completes the signing process successfully only if certain bit of signature equals certain bit of signing key.

malicious smart-card:

$$k \overset{U}{\leftarrow} \mathbb{Z}_q$$

$$r = kP.x$$

$$s = ke + dr$$

If $s_0 = d_{r_0}$

return (r, s)

return error

s_0 – the lowest bit of s

r_0 – the lowest byte of r

d_i – the i-th bit of d

CRYPTOPRO

Outline

1. Motivation
2. Related work
3. Blind signatures
4. Our contribution
5. Open problems

CRYPTOPRO

(Conventional) signature scheme

Sig (Signature) scheme:

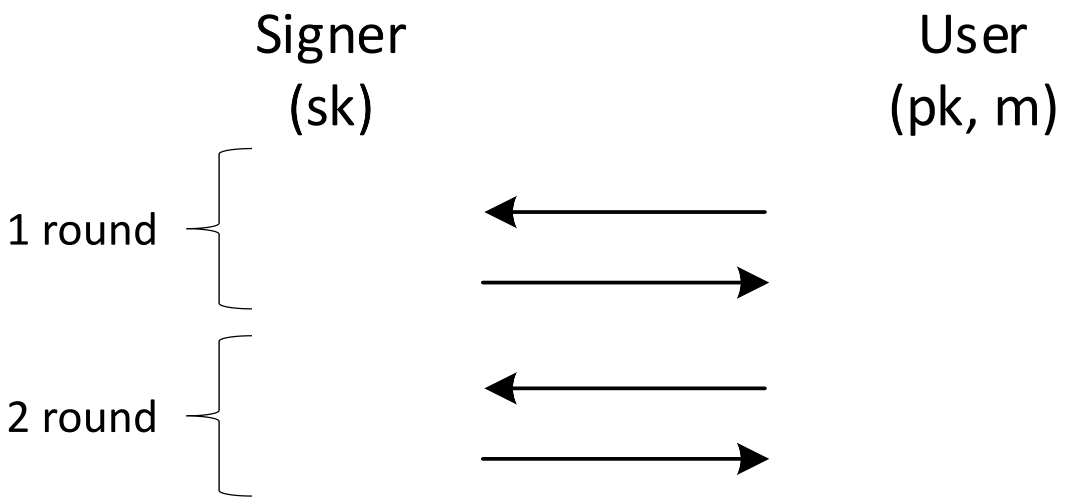
- $(sk, pk) \leftarrow \mathrm{Sig.KGen}(\,)$: key generation algorithm
- $\sigma \leftarrow \mathrm{Sig.Sign}(sk, m)$: signing algorithm
- $b \leftarrow \mathrm{Sig.Vf}\,(pk, m, \sigma)$: signature verification algorithm

(Standard) security notion: unforgeability under chosen message attack ($\mathrm{UF\text{-}CMA}$)

Blind signature scheme

BS (Blind Signature) scheme:

- $(sk, pk) \leftarrow \mathrm{BS.KGen}(\,)$: key generation algorithm
- $(b', \sigma) \leftarrow \langle \mathrm{BS.Signer}(sk), \mathrm{BS.User}(pk, m) \rangle$: interactive signing protocol
- $b \leftarrow \mathrm{BS.Vf}(pk, m, \sigma)$: signature verification algorithm



Blind version of signature

Definition

The BS scheme is a *blind version* of the Sig scheme, if the KGen and Vf algorithms of these schemes coincide and for any (sk, pk), any message m and any signature σ

$$\Pr[(1, \sigma) \leftarrow \langle \mathrm{BS.Signer}(sk), \mathrm{BS.User}(pk, m) \rangle] = \Pr[\sigma \leftarrow \mathrm{Sig.Sign}(sk, m)]$$

where the corresponding probability spaces are determined by the randomness used in the signing protocol and signing algorithm.

$\text{UF-CMA of Sig} \leftrightarrow \text{UF-CMA of BS}$

CRYPTOPRO

Example

GOST signature scheme

Sign(d, m)

$k \overset{U}{\leftarrow} \mathbb{Z}_q^*$

$R \leftarrow kP$

$e \leftarrow H(m)$

$r \leftarrow R.x \bmod q$

$s \leftarrow ke + dr$

Camenisch scheme is a blind version of GOST signature scheme

Camenisch blind signature scheme *

Signer(d)		**User**(Q, m)
$k \overset{\mathcal{U}}{\leftarrow} \mathbb{Z}_q^*$		
$R \leftarrow kP$	$\xrightarrow{R}$	
		$\alpha, \beta \overset{\mathcal{U}}{\leftarrow} \mathbb{Z}_q^*$
		$R' \leftarrow \alpha R + \beta P$
		$r' \leftarrow R'.x \bmod q$
		$e' \leftarrow H(m)$
		$r \leftarrow R.x \bmod q$
	$\xleftarrow{e}$	$e \leftarrow \alpha e' r (r')^{-1}$
$r \leftarrow R.x \bmod q$		
$s \leftarrow ke + dr$	$\xrightarrow{s}$	
		$s' \leftarrow s r' r^{-1} + \beta e'$
		$\sigma \leftarrow (r', s')$

*Camenisch J., Piveteau J., Stadler M. Blind signatures based on the discrete logarithm problem, 1994.

Blind signature

Security notions

unforgeability

correct signature can be generated only during the successful interaction with Signer

active adversary - User

blindness

no way to link the (message, signature) pair to the certain execution of the signing protocol

active adversary - Signer

Blind signature

Security notions

unforgeability

correct signature can be generated only during the successful interaction with Signer

active adversary - User

blindness

no way to link the (message, signature) pair to the certain execution of the signing protocol

active adversary - Signer

Sessions setting:

- Parallel
- Sequential

Signing key:

- Chosen by the adversary
- Honestly generated

Blind signature

Security notions

unforgeability

correct signature can be generated only during the successful interaction with Signer

active adversary - User

blindness

no way to link the (message, signature) pair to the certain execution of the signing protocol

active adversary - Signer

Sessions setting:

- Parallel
- Sequential

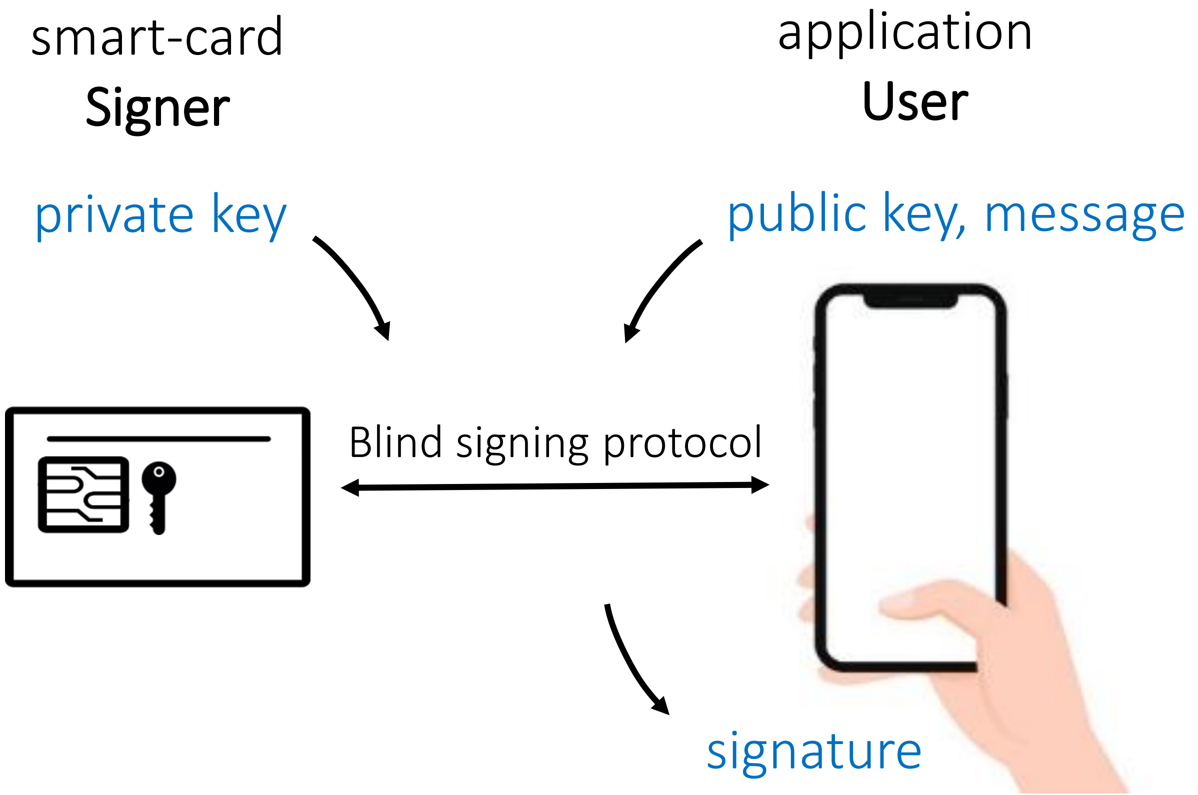
Signing key:

- Chosen by the adversary
- Honestly generated (wBL)

Outline

CRYPTOPRO

To use blind signature



Design rationale: Due to the blindness property, the malicious smart-card learns no information about the signature value, i.e. cannot «control» it.

Our contribution

Formal description of two new security notions for BS:

- **Weak unforgeabilty** (security against external adversary, wUNF):
  unforgeability against honest-but-curious (passive) User

- **Backdoor resilience** (security against adversary with agent, BDres):
  unforgeability in presence of backdoors in Signer

Our contribution

- **Weak unforgeabilty** (wUNF):
  unforgeability against honest-but-curious User

$$\mathbf{Exp}_{\mathsf{BS}}^{\mathrm{wUNF}}(\mathcal{A})$$

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$$
$$\mathcal{L} \leftarrow \varnothing$$
$$(m, \sigma) \leftarrow \mathcal{A}^{Sign}(\mathsf{pk})$$
$$\textbf{if } (m, \sigma) \in \mathcal{L} : \textbf{return } 0$$
$$\textbf{return } \mathsf{BS.Vf}(\mathsf{pk}, m, \sigma)$$

$$\mathit{Oracle\ Sign}(m)$$

$$(1, (\sigma; \boxed{view})) \leftarrow \langle \mathsf{BS.Signer}(\mathsf{sk}), \mathsf{BS.User}(\mathsf{pk}, m)\rangle$$
$$\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$$
$$\textbf{return } \sigma, \boxed{view}$$

Our contribution

- **Backdoor resilience** (BDres):
  unforgeability in presence of backdoors in Signer

$\mathbf{Exp}_{\mathsf{BS}}^{\mathrm{BDres}_k}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{BS.KGen}()$

$\mathcal{L} \leftarrow \varnothing$

$\mathsf{lost} \leftarrow \mathsf{false}$

$st \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk})$

$(m, \sigma) \xleftarrow{\$} \mathcal{A}_2^{Sign}(\mathsf{pk})$

$\textbf{if } ((m, \sigma) \in \mathcal{L}) \vee (\mathsf{lost} = \mathsf{true}):$

$\quad \textbf{return } 0$

$\textbf{return } \mathsf{BS.Vf}(\mathsf{pk}, m, \sigma)$

$\mathit{Oracle\ Sign}(m)$

$i \leftarrow 0$

$\textbf{do}$

$\quad (st, \sigma) \leftarrow \langle \mathcal{A}_1(st), \mathsf{BS.User}(\mathsf{pk}, m) \rangle$

$\quad i \leftarrow i + 1$

$\textbf{until } (i \geqslant k) \vee (\sigma \neq \perp)$

$\textbf{if } \sigma = \perp:$

$\quad \mathsf{lost} \leftarrow \mathsf{true}$

$\quad \textbf{return } \perp$

$\mathcal{L} \leftarrow \mathcal{L} \cup \{(m, \sigma)\}$

$\textbf{return } \sigma$

CRYPTOPRO

Our contribution

Theorem 1 (informal). If the BS scheme is wBL- and $\mathrm{UF\text{-}CMA}$-secure, then it is BDres-secure.

$$\mathrm{wBL} + \mathrm{UF\text{-}CMA} \rightarrow \mathrm{BDres}$$

CRYPTOPRO

Applying results to GOST

The Camenisch blind signature scheme is

✓ the blind version of GOST: $\mathbf{UF\text{-}CMA}$ of $\mathbf{Sig} \leftrightarrow \mathbf{UF\text{-}CMA}$ of $\mathbf{BS}$;

CRYPTOPRO

Applying results to GOST

The Camenisch blind signature scheme is

- ✓ the blind version of GOST: $\mathrm{UF\text{-}CMA}$ of $\mathrm{Sig} \leftrightarrow \mathrm{UF\text{-}CMA}$ of BS;
- ✓ perfectly wBL-secure (proven by Camenisch);

CRYPTOPRO

Applying results to GOST

The Camenisch blind signature scheme is

- ✓ the blind version of GOST: $\mathrm{UF\text{-}CMA}$ of $\mathrm{Sig} \leftrightarrow \mathrm{UF\text{-}CMA}$ of BS;
- ✓ perfectly wBL-secure (proven by Camenisch);

} $\mathrm{UF\text{-}CMA}$ of $\mathrm{Sig} \rightarrow \mathrm{BDres}$ of BS (by Theorem 1)

Applying results to GOST

The Camenisch blind signature scheme is

- ✓ the blind version of GOST: $\mathrm{UF\text{-}CMA}$ of $\mathrm{Sig} \leftrightarrow \mathrm{UF\text{-}CMA}$ of BS;
- ✓ perfectly wBL-secure (proven by Camenisch);

} $\mathrm{UF\text{-}CMA}$ of $\mathrm{Sig} \rightarrow \mathrm{BDres}$ of BS (by Theorem 1)

- ✓ wUNF-secure if GOST is $\mathrm{UF\text{-}CMA}$-secure (proven in our work).

Applying results to GOST

The Camenisch blind signature scheme is

- ✓ the blind version of GOST: $\mathrm{UF\text{-}CMA}$ of $\mathrm{Sig} \leftrightarrow \mathrm{UF\text{-}CMA}$ of BS;
- ✓ perfectly wBL-secure (proven by Camenisch);
- ✓ wUNF-secure if GOST is $\mathrm{UF\text{-}CMA}$-secure (proven in our work).

(first two items together:) $\mathrm{UF\text{-}CMA}$ of $\mathrm{Sig} \rightarrow \mathrm{BDres}$ of BS (by Theorem 1)

Thus, if GOST is $\mathrm{UF\text{-}CMA}$-secure, then the Camenisch scheme is BDres- and wUNF-secure.

Applying results to GOST

The Camenisch blind signature scheme is

- ✓ the blind version of GOST: $\mathbf{UF\text{-}CMA}$ of $\mathbf{Sig} \leftrightarrow \mathbf{UF\text{-}CMA}$ of $\mathbf{BS}$;
- ✓ perfectly $\mathbf{wBL}$-secure (proven by Camenisch);
- ✓ $\mathbf{wUNF}$-secure if GOST is $\mathbf{UF\text{-}CMA}$-secure (proven in our work).

(first two items) } $\mathbf{UF\text{-}CMA}$ of $\mathbf{Sig} \rightarrow \mathbf{BDres}$ of $\mathbf{BS}$ (by Theorem 1)

Thus, if GOST is $\mathbf{UF\text{-}CMA}$-secure, then the Camenisch scheme is $\mathbf{BDres}$- and $\mathbf{wUNF}$-secure.

What does it mean for practice?

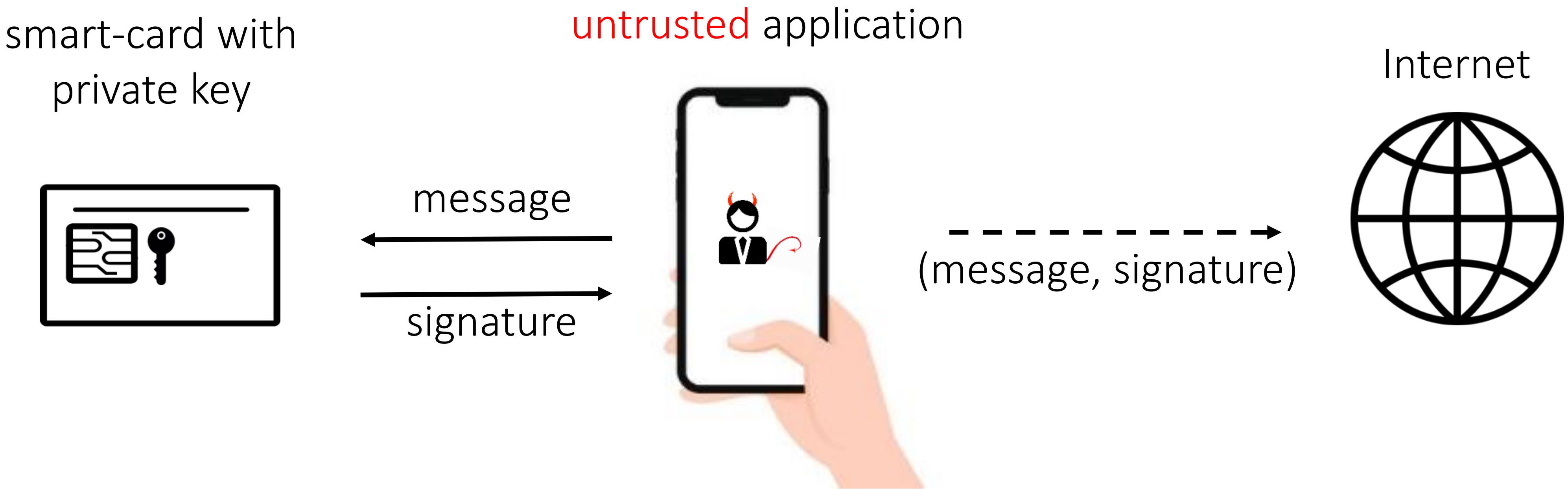
In order to provide security against **backdoors in smart-cards** and **memory leak in application** in case of using GOST, it is enough to implement the Camenisch blind signature scheme.

Outline

1. Motivation
2. Related work
3. Blind signatures
4. Our contribution
5. Open problems

CRYPTOPRO

Extending security notion



Fully active adversary on the application side (untrusted application or no password-based protection in case of smart-card theft).

Goal: to make a forgery, in particular, by key recovery

Open problems

The Camenisch blind signature scheme and unforgeability with **active adversary**:

- is not secure in parallel sessions setting

  CTCrypt'2022 "On the (im)possibility of secure ElGamal blind signatures"

- **potentially secure** in sequential sessions setting (enough for smart-card case)

  positive results for the Schnorr blind signature and its modifications

CRYPTOPRO

Thank you for your attention!

lah@cryptopro.ru