# Security of Okamoto Identification Scheme - a Defense against Ephemeral Key Leakage and Setup

Łukasz Krzywiecki, Mirosław Kutyłowski

Department of Computer Science
Faculty of Fundamental Problems of Technology
Wrocław University of Science and Technology

The Fifth International Workshop on Security in Cloud Computing
SCC 2017, Abu Dhabi, UAE

# Identification

## Identification Scheme (IS)

a scheme involving two parties:

- prover – proves his identity,
- verifier – accepts or rejects the proof

## Attribures of the authenticator

- what the prover has (key, token, etc.),
- what the prover knows (secret key, password, etc.),
- what the prover are (e.g. biometric)

We concentrate on *"what the authenticator knows"* methodology.

# Some known schemes

## Some dedicated construction


Schnorr, C.P.:
Efficient signature generation by smart cards.
J. Cryptology **4**(3) (1991) 161–174

Fiat, A., Shamir, A.:
How To Prove Yourself: Practical Solutions to Identification and Signature Problems.

Feige, U., Fiat, A., Shamir, A.:
Zero-knowledge proofs of identity.

Guillou, L.C., Quisquater, J.J.:
A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory.

Okamoto, T.:
Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes.

Kurosawa, K., Heng, S.H.:
Identity-Based Identification Without Random Oracles

Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.:
Resettable zero-knowledge (extended abstract).

Bellare, M., Fischlin, M., Goldwasser, S., Micali, S.:
Identification Protocols Secure against Reset Attacks.

# General Construction

## Asymmetric cryptography setup

- the prover has a long term **secret key**
- the verifier has the corresponding **public key**

## Zero Knowledge Proof

- the verifier is convinced,
- gets no information about the prover's secret.

# General Construction

## Three rounds

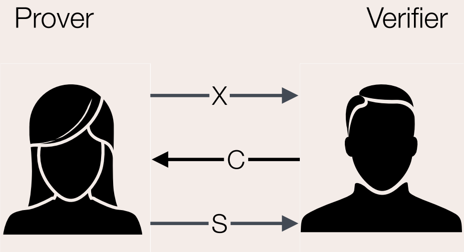
- **commitment** the prover sends a commitment to some random ephemeral value.
- **challenge** the verifier random unpredictable challenge.
- ***response*** the prover sends the result of some computations over the challenge, the secret and the ephemeral value .

## Verification

The prover is accepted if the response ”agrees” with the computation involving the commitment, the challenge, the response and the public key of the prover.

# General Construction

## Protocol




- $x$ for commitment
- $c$ for challenge
- $s$ for proof

# Deniability

## Deniable Identification

**Simulatability**: The prover without the secret key can produce the transcript itself.

## Distinguisher

Cannot tell

- whether the transcript was a result of the regular protocol execution.
- or the transcript was simulated.

even if it was given the secret key.

# Okamoto identification scheme

## Initialization Stage

params $\leftarrow$ ParGen($1^\lambda$): Let $\mathbb{G} = (p, q, g, G) \leftarrow \mathcal{G}(1^\lambda)$, s.t. DL assumption holds. Set params $= (p, q, g_1, g_2, G)$.

KeyGen(): sk = $a_1, a_2 \leftarrow \mathbb{Z}_q^*$, pk $= A = g_1^{a_1} g_2^{a_2}$. Output (sk, pk).

Figure: The Okamoto identification scheme.

# Okamoto identification scheme



## Operation Stage

$\pi(\mathcal{P}(a_1, a_2), \mathcal{V}(A))$:

1. $\mathcal{P}$: $x_1, x_2 \in_R \mathbb{Z}_q^*$, $X = g_1^{x_1} g_2^{x_2}$
   sends $X$ to the verifier $\mathcal{V}$.
2. $\mathcal{V}$ : $c \in_R \mathbb{Z}_q^*$,
   sends $c$ to the prover $\mathcal{P}$.
3. $\mathcal{P}$ : $s_1 = x_1 + a_1 c$ $s_2 = x_2 + a_2 c$
   sends $s_1, s_2$ to the verifier $\mathcal{V}$.

## Verifier accepts the Prover iff

$$g_1^{s_1} g_2^{s_2} == X A^c$$

# Okamoto identification scheme



## Operation Stage

| $\mathcal{P}(a_1, a_1)$ | | $\mathcal{V}(A = g_1^{a_1} g_2^{a_2})$ |
|---|---|---|
| $x_1, x_2 \in_R \mathbb{Z}_q^*$, $X = g_1^{x_1} g_2^{x_2}$ | | |
| | $\xrightarrow{X}$ | |
| | | $c \in_R \mathbb{Z}_q^*$ |
| | $\xleftarrow{c}$ | |
| $s_1 = x_1 + a_1 c$, $s_2 = x_2 + a_2 c$ | $\xrightarrow{s_1, s_2}$ | |
| | | Accept iff $g_1^{s_1} g_2^{s_2} = XA^c$ |

**Protocol Simulation**

1. Simulator chooses $\tilde{s}_1, \tilde{s}_2, \tilde{c}$ first
2. then $\tilde{X} = (g_1^{\tilde{s}_1} g_2^{\tilde{s}_2} / A^{\tilde{c}})$.

The tuples
$T = (X, c, s_1, s_2)$ - from the protocol execution
$\tilde{T} = (\tilde{X}, \tilde{c}, \tilde{s}_1, \tilde{s}_2)$ - simulated
are identically distributed.

# Device based authentication

## Device

Small hardware which *securely* store the authentication keys inside (e.g smartcards).

## Adversaries Attacks

- tries to extract what was put inside,
- tries to manipulate what is inside,
- ...

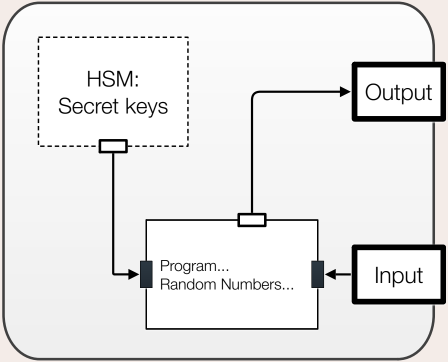
Common threats:

- invasive attack,
- power analysis,
- emission of radiation,
- ...

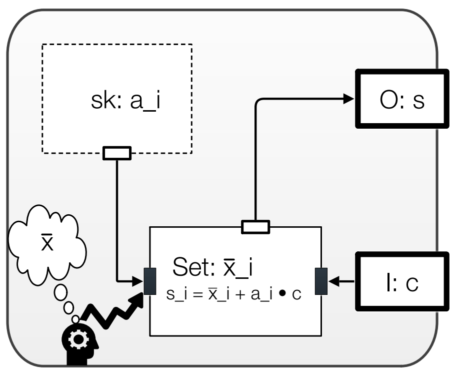
# Okamoto identification scheme

## Deniability

### Device architecture



### Attack - subliminal setting of ephemerals




Okamoto IS is not secure if $\bar{x}$ is known to the adversary.
$\mathcal{A}$ can easily compute the secret key $a_i = (s_i - \bar{x}_i)/c$.

# Chosen Prover Ephemeral

## Security experiment

The experiment $\mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE},\lambda,\ell}$:

**Init stage** $\mathsf{params} \leftarrow \mathsf{ParGen}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}()$.
$\mathcal{A} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ given the public key $\mathsf{pk}$.

**Query stage** $\mathcal{A}$ runs a polynomial number $\ell$ of
$\pi(\mathcal{P}^{\bar{x}_i}(\mathsf{sk}, \mathsf{pk}), \tilde{\mathcal{V}}(\mathsf{pk}, \bar{x}_i))$
collecting view $\mathsf{v}^{\mathcal{P},\tilde{\mathcal{V}},\vec{\bar{x}}(\ell)}$,
where $\bar{x}_i \in \{\bar{x}_1, \ldots, \bar{x}_\ell\}$ are injected

**Impersonation stage** $\mathcal{A}$ runs the protocol
$\pi(\tilde{\mathcal{P}}(\mathsf{pk}, \mathsf{v}^{\mathcal{P},\tilde{\mathcal{V}},\vec{\bar{x}}(\ell)}), \mathcal{V}(\mathsf{pk}))$

# Chosen Prover Ephemeral



## Adversary advantage

The advantage of $\mathcal{A}$ in the experiment $\mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE},\lambda,\ell}$ as **probability of acceptance** in the *impersonation stage*:

$$\mathbf{Adv}(\mathcal{A}, \mathtt{Exp}_{\mathsf{IS}}^{\mathsf{CPE},\lambda,\ell}) = \Pr[\pi(\tilde{\mathcal{P}}(\mathsf{pk}, \mathsf{v}^{\mathcal{P},\tilde{\mathcal{V}},\vec{x}(\ell)}), \mathcal{V}(\mathsf{pk})) \to 1].$$

The identification scheme is secure if it is negligible in $\lambda$.

## Security of identification scheme

$\mathcal{A}$ **probability of acceptance** is negligible in $\lambda$.

# Solution

## Bilinear Map

Let $G_T$ be another group of a prime order $q$. We assume that $\hat{e}: G \times G \to G_T$ is a bilinear map s.t. following condition holds:

1) *Bilinearity*: $\forall a, b \in \mathbb{Z}_q^*, \forall g, g \in G$: $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$.
2) *Non-degeneracy*: $\hat{e}(g, g) \neq 1$.
3) *Computability*: $\hat{e}$ is efficiently computable.

## New generator

Let $\mathcal{H}: \{0, 1\}^* \to G$ be a hash function.
We compute another element of $G$ denoted by $\hat{g}$.

# Modified Okamoto identification scheme



## Operation Stage

| $\mathcal{P}(a_1, a_2)$ | | $\mathcal{V}(A = g_1^{a_1} g_2^{a_2})$ |
|---|---|---|
| $x_1, x_2 \in_R \mathbb{Z}_q^*$, | | |
| $X = g_1^{x_1} g_2^{x_2}$ | $\xrightarrow{X}$ | |
| | $\xleftarrow{c}$ | $c \in_R \mathbb{Z}_q^*$ |
| $\hat{g} = \mathcal{H}(X\|c)$ | | |
| $S_1 = \hat{g}^{x_1 + a_1 c}$, | | $\hat{g} = \mathcal{H}(X\|c)$ |
| $S_2 = \hat{g}^{x_2 + a_2 c}$ | $\xrightarrow{S_1, S_2}$ | Accept iff |
| | | $\hat{e}(S_1, g_1) \cdot \hat{e}(S_2, g_2) =$ |
| | | $= \hat{e}(\hat{g}, X \cdot A^c)$ |

## Protocol Simulation for Passive Adversary

1. Simulator chooses $\tilde{s}_1, \tilde{s}_2, \tilde{c}$ first
2. $\tilde{X} = (g_1^{\tilde{s}_1} g_2^{\tilde{s}_2} / A^{\tilde{c}})$.
3. $\hat{g} = \mathcal{H}(\tilde{X}|\tilde{c})$
4. $\tilde{S}_1 = \hat{g}^{\tilde{s}_1}$, $\tilde{S}_2 = \hat{g}^{\tilde{s}_2}$

The tuples
$T = (X, c, S_1, S_2)$ - from the protocol execution
$\tilde{T} = (\tilde{X}, \tilde{c}, \tilde{S}_1, \tilde{S}_2)$ - simulated
are identically distributed.

# Security Experiment

## Init stage

### CDH Breaking

1. given CDH$(g, g^{\alpha}, g^{\beta})$
2. set $A = g^{\alpha}$
3. set $a_2, \omega \leftarrow_R \mathbb{Z}_q^*$,
4. set $g_1 = g$, $g_2 = g^{\omega}$
5. we have $g_1^{a_1} = A / g_2^{a_2}$

We simulate Query stage in ROM. We use rewinding technique

# Security Experiment

## Query stage

### Protocol Simulation for Active Adversary

1. **ROM table $\mathcal{O}_{\mathcal{H}}$:** Three columns $I, H, r$: for the input, the output and the masked exponent respectively.
2. **New query:** $r_i \leftarrow_R \mathbb{Z}_q^*$, compute $H_i = g^{r_i}$, insert $(I_i, H_i, r_i)$, return $H_i$.
   **Commitment:** When injected ephemeral $\bar{x}_1$, $\bar{x}_2$ compute $\tilde{X} = g_1^{\bar{x}_1} g_2^{\bar{x}_2}$ and send $\bar{X}$ to the verifier
   **Proof:** On receiving $\tilde{c}$, call $\mathcal{O}_{\mathcal{H}}(\bar{X}|\tilde{c})$, locate and retrieve the corresponding $g^r$ and $r$. We set $\hat{g} = g^r$.
   Compute:
   $\tilde{S}_1 = (g_1{}^{x_1})^r (A/g_2^{a_2})^{rc} = \hat{g}^{\bar{x}_1 + a_1 c}$
   $\tilde{S}_2 = (g_2{}^{x_2})^r (g_2^{a_2})^{rc} = \hat{g}^{\bar{x}_2 + a_2 c}$

Verification holds. $T = (X, c, S_1, S_2)$, and $\tilde{T} = (\tilde{X}, \tilde{c}, \tilde{S}_1, \tilde{S}_2)$ identically distributed.

# Security Experiment
## Impersonation stage



Then we use the *rewinding technique*:

1. we run protocol twice for
2. the same fixed commitment $X$,
3. use different challenges $c, c'$
4. in ROM inject $g^\beta$
5. get responses $S_1, S_2$, and $S'_1, S'_2$.
6. two resulting tuples $(X, c, S_1, S_2)$, $(X, c', S'_1, S'_2)$
7. these enable us to break the underlying $\mathrm{GDH}(g, g^\alpha, g^\beta)$.

# Security rationale

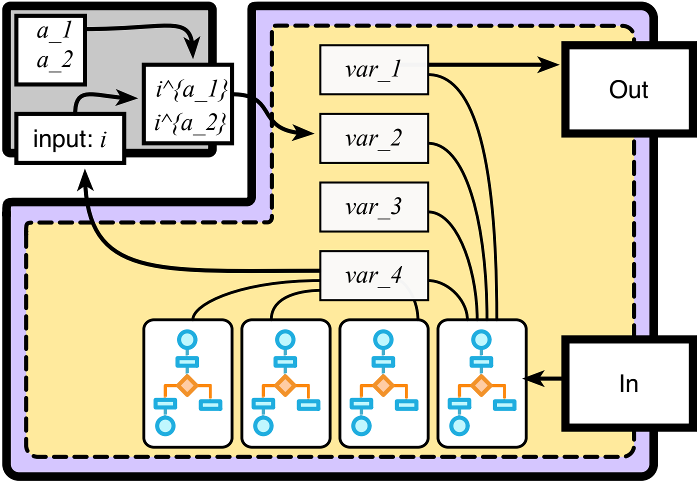


## Problems for the Adversary

1. from $\hat{g}^{\bar{x}+ac}$ it is hard to get $a$
2. if you know $\bar{x}, c$ you can compute $\hat{g}^a$
3. knowing $\hat{g}_1^{a_1}, \ldots, \hat{g}_\ell^{a_1}$
   still it is hard to compute $\hat{g}_n^{a_1}$
   for completely new element $\hat{g}_n$
4. knowing $\hat{g}_1^{a_2}, \ldots, \hat{g}_\ell^{a_2}$
   still it is hard to compute $\hat{g}_n^{a_2}$
   for completely new element $\hat{g}_n$

# Shifting computations to Cloud

## Architecture



# Shifting computations to Cloud
## Possible Advantages



### "Gray" Secure Module

1. user retain "Gray" Secure Module
2. "Gray" Secure Module – black box

### "Yellow" Insecure Module

1. yellow part can be outsourced to cloud
2. yellow part – white box

### Adversary cloud cannot:

- extract long term secret keys,
- impersonate user

# Thanks



Thank You