FORMAL ANALYSIS OF COMPARISON-BASED NON-MALLEABILITY FOR COMMITMENTS

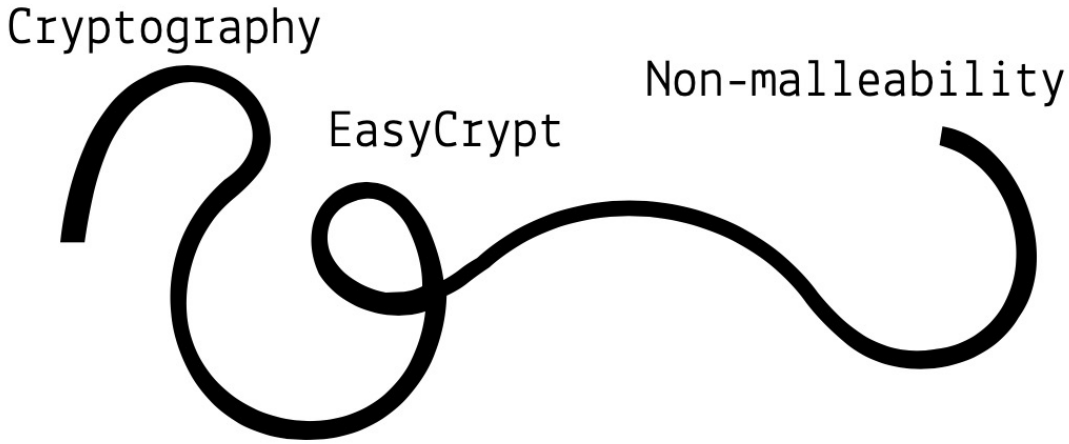
Denis Firsov, Sven Laur, and Ekaterina Zhuchko

TAL TECH

UNIVERSITY OF TARTU
1632

QUICK OVERVIEW

- Background on cryptography.
- Background on EasyCrypt.
- Non-malleability.
- Formal analysis of non-malleability.

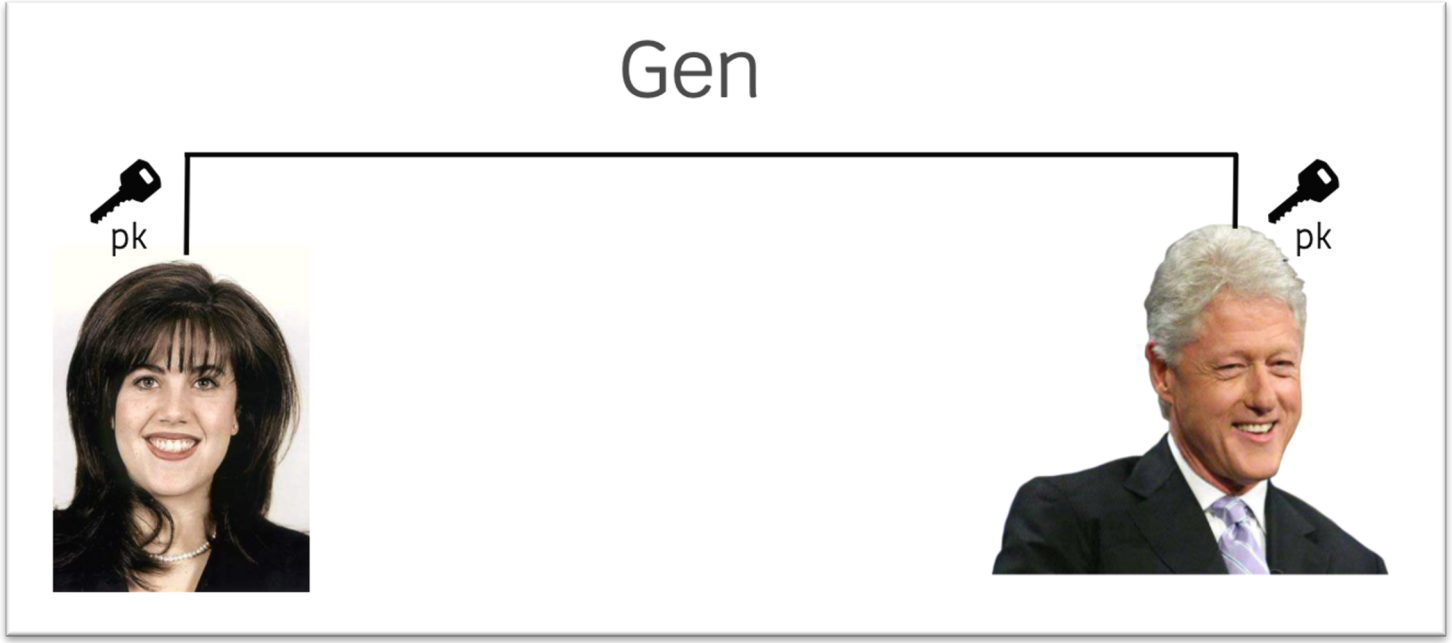


COMMITMENT SCHEMES

- Key generation process.
- Two participating parties: Sender and Receiver.
- Two-phase protocol: Commit and Reveal.
- Sender has a private message.
- Sender commits to a message and sends it to the Receiver.
- At a later stage, the Receiver checks the authenticity of the message.

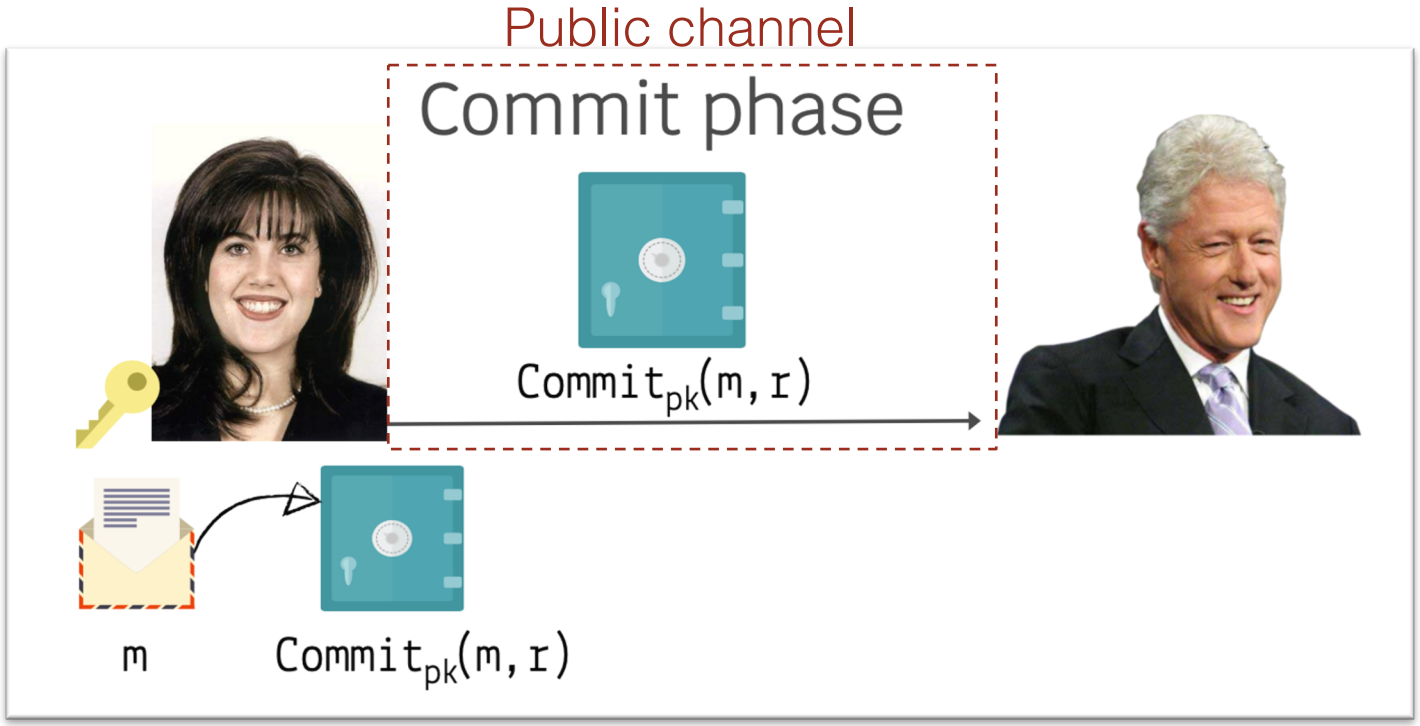
COMMITMENT SCHEMES

- Key generation process: Gen.
- Public key is generated and distributed amongst the users.



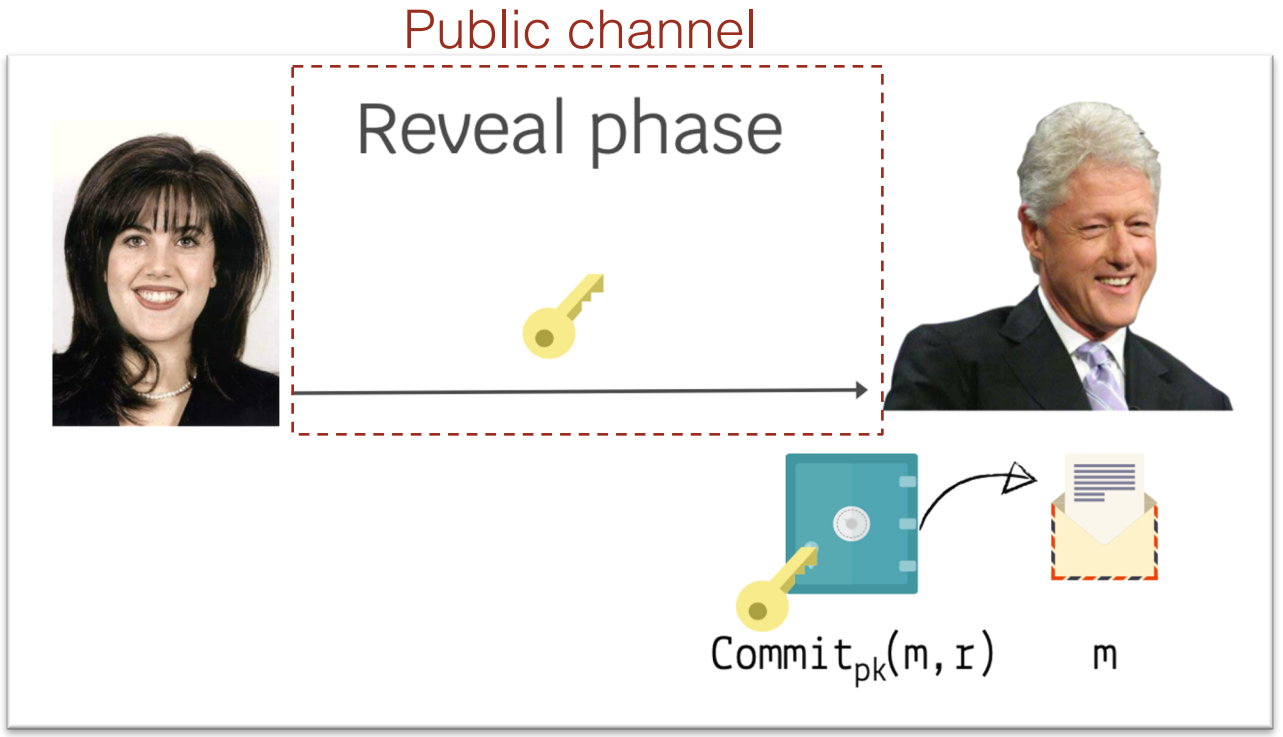
COMMITMENT SCHEMES

- Phase 1: Commit.
- Sender commits to a message.



COMMITMENT SCHEMES

- Phase 2: Reveal.
- At a later stage, the Receiver checks the authenticity of the message.



COMMITMENT SCHEMES

- Randomised public key generation function $Gen: Rand \rightarrow Pub\_key$.
- A commitment function $Commit_{pk}: \mathcal{M} \times Rand \rightarrow \mathcal{C} \times \mathcal{D}$.
- A verification function $Verify_{pk}: \mathcal{M} \times \mathcal{C} \times \mathcal{D} \rightarrow \{True, False\}$.

COMMITMENT SCHEMES – SECURITY PROPERTIES

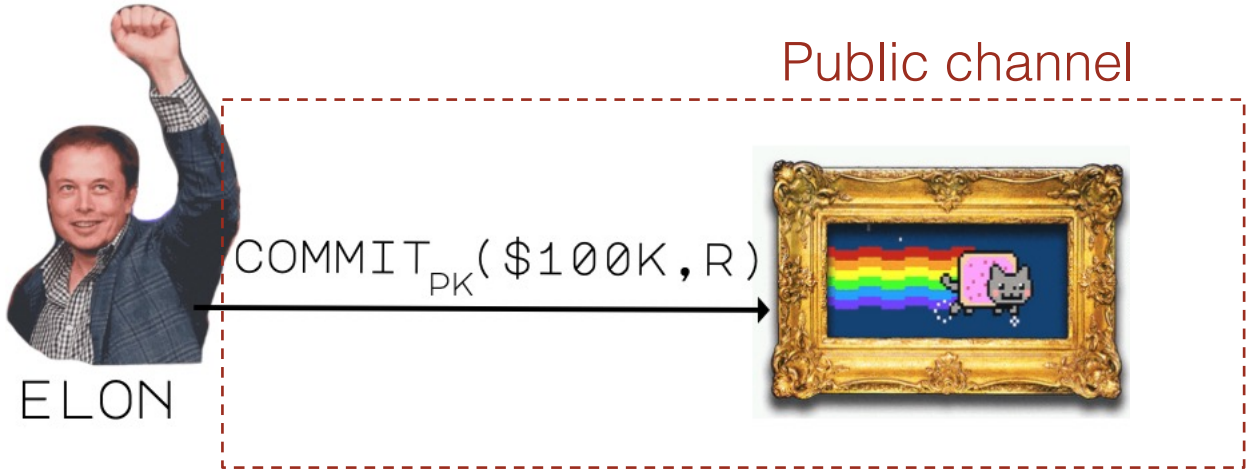
- Hiding: Receiver cannot see the message inside the commitment.
- Binding: Sender is bound to the committed message.
- Non-malleability.

MOTIVATING EXAMPLE 1: NFT AUCTION

Public channel

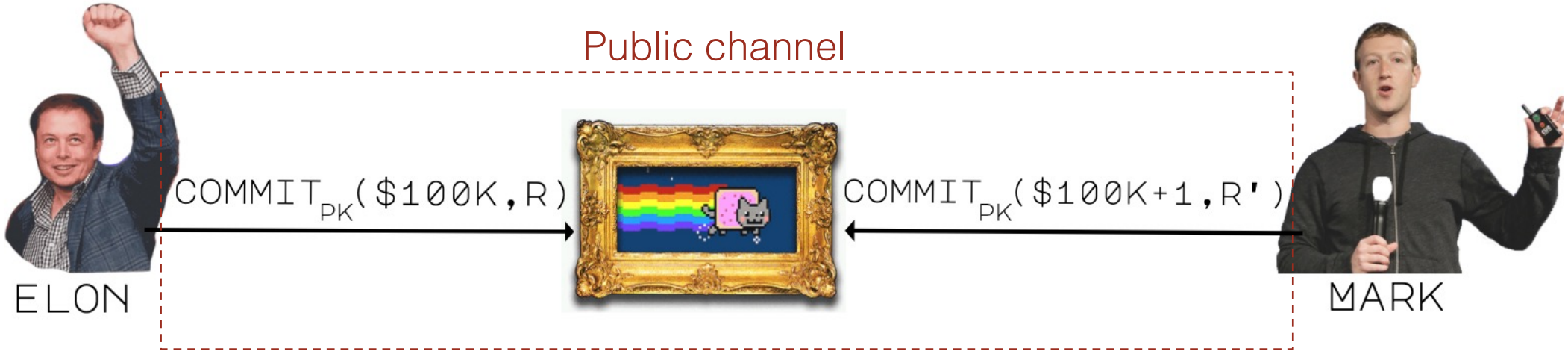
- A new item appears in a sealed-bid auction.

MOTIVATING EXAMPLE 1: NFT AUCTION



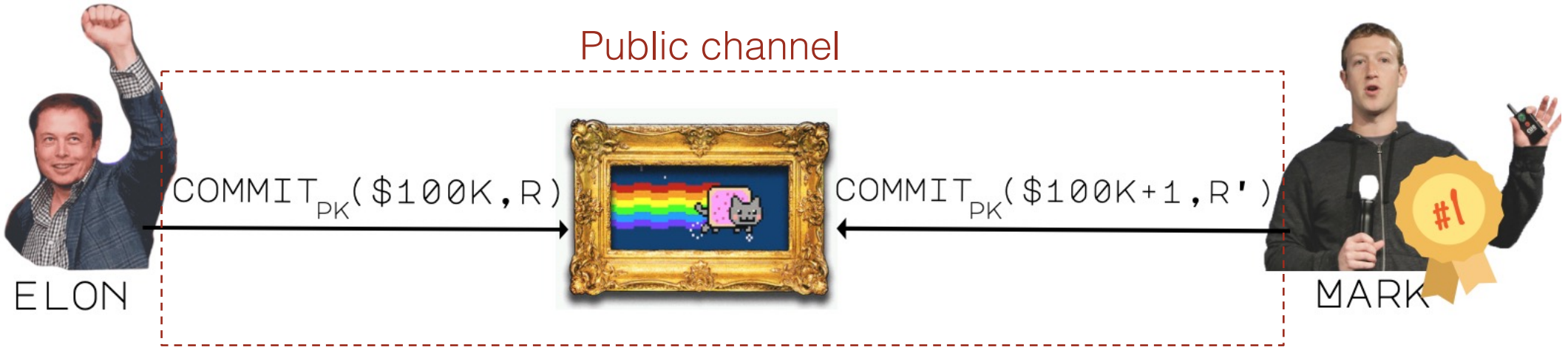
- Elon is honest and places his bid sealed inside the commitment.

MOTIVATING EXAMPLE 1: NFT AUCTION



- Mark is dishonest.
- He cannot see $\$100K$ but constructs his bid based on $\mathbf{Commit_{pk}(\$100K, R)}$.

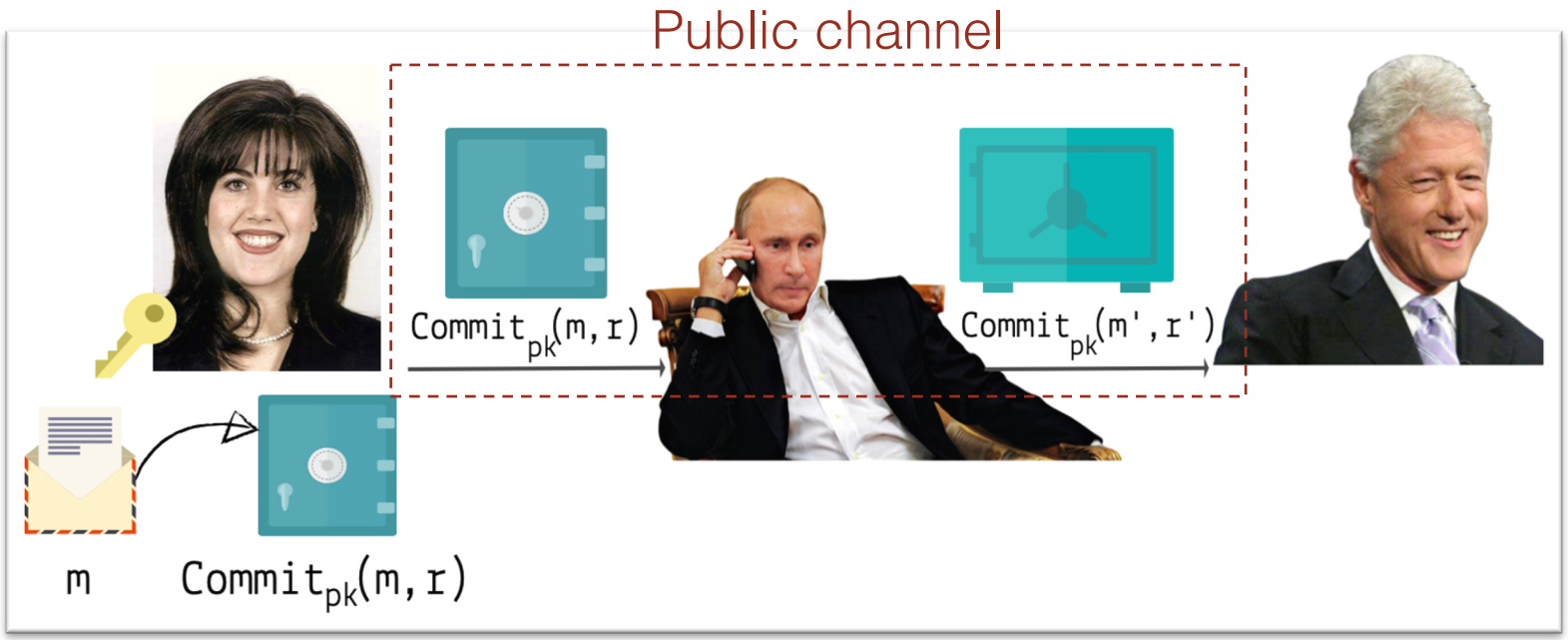
MOTIVATING EXAMPLE 1: NFT AUCTION



- This scenario is possible due to malleability.

MOTIVATING EXAMPLE 2: MAN-IN-THE-MIDDLE ATTACK

- We have an active adversary who can modify commitments.

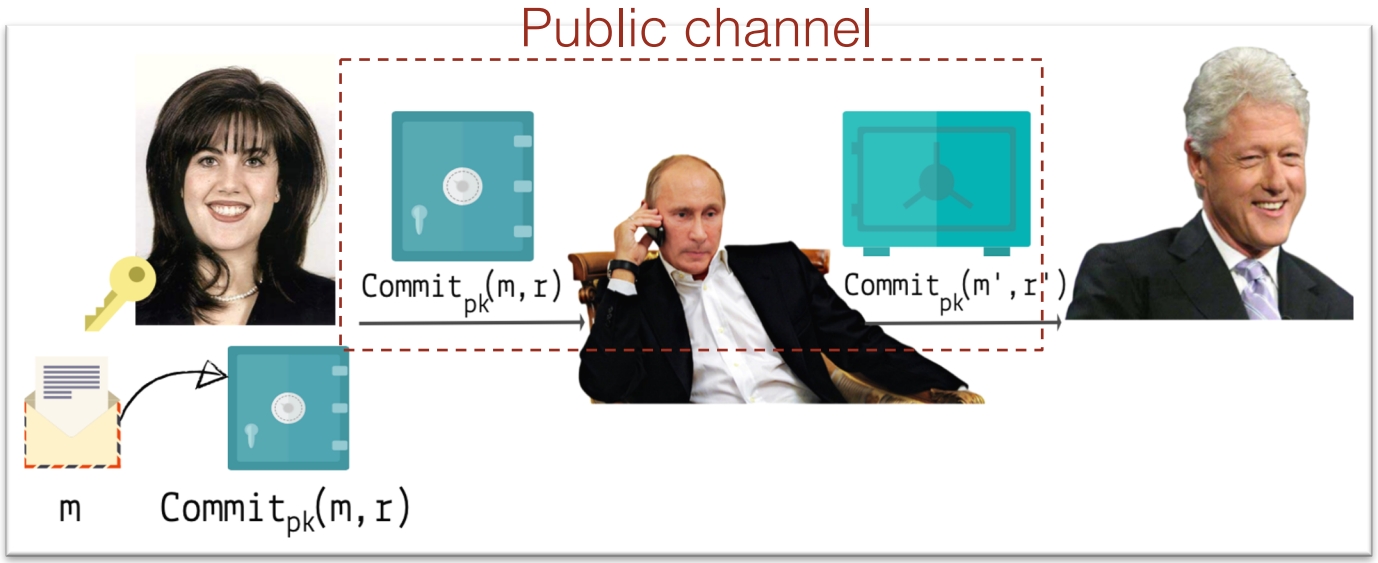


MOTIVATING EXAMPLE 2: MAN-IN-THE-MIDDLE ATTACK

- $\mathbf{m}'$ has to be non-trivially related to the original message m.

MOTIVATING EXAMPLE 2: MAN-IN-THE-MIDDLE ATTACK

- **Goal**: $\mathbf{Commit_{pk}(m, r)}$ does not help the adversary to create other commitments $\mathbf{Commit_{pk}(m', r')}$



EASYCRYPT

- Proof assistant for reasoning about adversarial code.
- Probabilistic relational Hoare logic.
- Module system for structuring cryptographic constructions.
- Concrete and abstract modules (to model adversaries).

EASYCRYPT EXAMPLE

- Define the commitment scheme:

```
type pub_key, message, randomness, commitment, openingkey.

op Gen :  pub_key distr.
op Commit (pk :  pub_key) (m :  message, r :  randomness):  commitment * openingkey.
op Verify :  pub_key -> message * (commitment * openingkey) -> bool.

axiom correct pk :
       pk \in Gen => forall m, Verify pk (m, (Commit pk (m, r))).
```

EASYCRYPT EXAMPLE

- Define adversary who is attacking the hiding property of the commitment scheme:

```
module type Unhider = {
 proc choose(pk :  pub_key) :  message * message
 proc guess(c :  commitment) :  bool
}.
```

EASYCRYPT EXAMPLE

- Now, we can test this adversary in the hiding game:

```
module HidingExperiment (A : Unhider) = {
 proc main() :  bool = {
      var b, b', r, m0, m1, pk, c, d;

      pk <$ Gen;
      r <$ Rand;
      (m0, m1) <@ A.choose(pk);
      b <$ {0,1};
      (c, d) <- Commit pk (b ?  m1 :  m0) r;
      b' <@ A.guess(c);

      return (b = b');
 }
}.
```

EASYCRYPT EXAMPLE

- We now have an adversary, a commitment scheme and a hiding experiment.
- We can model the security with the following lemma:

```
lemma hiding:  forall (A <:  Unhider) &m,
      Pr[ HidingExperiment(A).main() @ &m :  res] = 1/2.
```

NON-MALLEABILITY DEFINITION

- Comparison-based non-malleability (Laur and Nyberg, 2007):

$\mathrm{GN}_0^{\mathcal{A}}$	$\mathrm{GN}_1^{\mathcal{A}}$
1 : pk $\leftarrow$ Gen	1 : pk $\leftarrow$ Gen
2 : $r \leftarrow\$$ Rand	2 : $r \leftarrow\$$ Rand
3 : $\mathcal{M} \leftarrow \mathcal{A}$.init(pk)	3 : $\mathcal{M} \leftarrow \mathcal{A}$.init(pk)
4 : $m \leftarrow\$ \mathcal{M}$	4 : $m \leftarrow\$ \mathcal{M}; \bar{m} \leftarrow\$ \mathcal{M}$
5 : $(c, d) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(m, r)$	5 : $(\bar{c}, \bar{d}) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(\bar{m}, r)$
6 : $(n, R(\cdot), \hat{c}_1, ..., \hat{c}_n) \leftarrow \mathcal{A}$.commit$(c)$	6 : $(n, R(\cdot), \hat{c}_1, ..., \hat{c}_n) \leftarrow \mathcal{A}$.commit$(\bar{c})$
7 : $((\hat{m}_1, \hat{d}_1), ..., (\hat{m}_n, \hat{d}_n)) \leftarrow \mathcal{A}$.decommit$(d)$	7 : $((\hat{m}_1, \hat{d}_1), ..., (\hat{m}_n, \hat{d}_n)) \leftarrow \mathcal{A}$.decommit$(\bar{d})$
8 : if $c \in \{\hat{c}_1, ..., \hat{c}_n\}$ **return** 0	8 : if $\bar{c} \in \{\hat{c}_1, ..., \hat{c}_n\}$ **return** 0
9 : $b \leftarrow \mathsf{Verify}_{\mathsf{pk}}(\hat{m}_i, \hat{c}_i, \hat{d}_i)\ \forall i \in \{1 \leq i \leq n\}$	9 : $b \leftarrow \mathsf{Verify}_{\mathsf{pk}}(\hat{m}_i, \hat{c}_i, \hat{d}_i)\ \forall i \in \{1 \leq i \leq n\}$
10 : **return** $R(m, \hat{m}_1, ..., \hat{m}_n)$	10 : **return** $R(m, \hat{m}_1, ..., \hat{m}_n)$

- Adversary wins if he can distinguish these two games:

$$\mathrm{Adv}_{\mathcal{C}}^{\mathrm{nmo}}(\mathcal{A}) = \left|\Pr\left[GN_0^{\mathcal{A}} = 1\right] - \Pr\left[GN_1^{\mathcal{A}} = 1\right]\right|$$

NON-MALLEABILITY DEFINITION

- Comparison-based non-malleability (Laur and Nyberg, 2007):

$\mathrm{GN}_0^{\mathcal{A}}$		$\mathrm{GN}_1^{\mathcal{A}}$	
1 :	$\mathsf{pk} \leftarrow \mathsf{Gen}$	1 :	$\mathsf{pk} \leftarrow \mathsf{Gen}$
2 :	$r \leftarrow\$ \mathsf{Rand}$	2 :	$r \leftarrow\$ \mathsf{Rand}$
3 :	$\mathcal{M} \leftarrow \mathcal{A}.\mathsf{init}(\mathsf{pk})$	3 :	$\mathcal{M} \leftarrow \mathcal{A}.\mathsf{init}(\mathsf{pk})$
4 :	$m \leftarrow\$ \mathcal{M}$	4 :	$m \leftarrow\$ \mathcal{M}; \bar{m} \leftarrow\$ \mathcal{M}$
5 :	$(c, d) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(m, r)$	5 :	$(\bar{c}, \bar{d}) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(\bar{m}, r)$
6 :	$(n, R(\cdot), \hat{c}_1, ..., \hat{c}_n) \leftarrow \mathcal{A}.\mathsf{commit}(c)$	6 :	$(n, R(\cdot), \hat{c}_1, ..., \hat{c}_n) \leftarrow \mathcal{A}.\mathsf{commit}(\bar{c})$
7 :	$((\hat{m}_1, d_1), ..., (\hat{m}_n, d_n)) \leftarrow \mathcal{A}.\mathsf{decommit}(d)$	7 :	$((\hat{m}_1, d_1), ..., (\hat{m}_n, d_n)) \leftarrow \mathcal{A}.\mathsf{decommit}(d)$
8 :	if $c \in \{\hat{c}_1, ..., \hat{c}_n\}$ **return** 0	8 :	if $\bar{c} \in \{\hat{c}_1, ..., \hat{c}_n\}$ **return** 0
9 :	$b \leftarrow \mathsf{Verify}_{\mathsf{pk}}(\hat{m}_i, \hat{c}_i, \hat{d}_i)\ \forall i \in \{1 \leq i \leq n\}$	9 :	$b \leftarrow \mathsf{Verify}_{\mathsf{pk}}(\hat{m}_i, \hat{c}_i, \hat{d}_i)\ \forall i \in \{1 \leq i \leq n\}$
10 :	**return** $R(m, \hat{m}_1, ..., \hat{m}_n)$	10 :	**return** $R(m, \hat{m}_1, ..., \hat{m}_n)$

- Adversary wins if he can distinguish these two games:

$$\mathrm{Adv}_{\mathcal{C}}^{\mathrm{nmo}}(\mathcal{A}) = \left|\Pr\left[GN_0^{\mathcal{A}} = 1\right] - \Pr\left[GN_1^{\mathcal{A}} = 1\right]\right|$$

COUNTER-EXAMPLE

- We define adversary who is attacking the non-malleability property of the commitment scheme:

$\mathcal{A}.\mathsf{init}(\mathsf{pk})$

$\mathcal{M} \leftarrow \{\frac{1}{2}\mathsf{true}, \frac{1}{2}\mathsf{false}\}$

return $\mathcal{M}$

$\mathcal{A}.\mathsf{commit}(\mathrm{c})$

$r \leftarrow\$ \mathsf{Rand}$

$(\hat{c}, \hat{d}) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(\mathsf{false}, \mathsf{r})$

$R \leftarrow \lambda m_0 m_1.(m_0 = \mathsf{false}) \wedge\ (m_1 = \mathsf{false})$

return $(1, R, \hat{c})$

$\mathcal{A}.\mathsf{decommit}(\mathrm{d})$

if $\mathsf{Verify}_{\mathsf{pk}}(\mathsf{false}, \mathsf{c}, \mathsf{d})$

return $(\mathsf{false}, \hat{d})$

else fail

- Now this adversary will attack the non-malleability games.

COUNTER-EXAMPLE

$\mathrm{GN}_0^{\mathcal{A}}$

1 : pk $\leftarrow$ Gen
2 : $r \leftarrow\$$ Rand
3 : $\mathcal{M} \leftarrow \mathcal{A}$.init(pk)
4 : $m \leftarrow\$ \mathcal{M}$
5 : $(c, d) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(m, r)$
6 : $(n, R(\cdot), \hat{c}_1, ..., \hat{c}_n) \leftarrow \mathcal{A}.\mathsf{commit}(c)$
7 : $((\hat{m}_1, \hat{d}_1), ..., (\hat{m}_n, \hat{d}_n)) \leftarrow \mathcal{A}.\mathsf{decommit}(d)$
8 : if $c \in \{\hat{c}_1, ..., \hat{c}_n\}$ **return** 0
9 : $b \leftarrow \mathsf{Verify}_{\mathsf{pk}}(\hat{m}_i, \hat{c}_i, \hat{d}_i)\ \forall i \in \{1 \leq i \leq n\}$
10 : **return** $R(m, \hat{m}_1, ..., \hat{m}_n)$

- $\Pr\left[GN_0^{\mathcal{A}} : m = false \ \wedge win\right] \approx \frac{1}{2}$.
- $\Pr\left[GN_0^{\mathcal{A}} : m = true \ \wedge win\right] = 0$.

$\mathcal{A}$.init(pk)

$\mathcal{M} \leftarrow \{\frac{1}{2}\mathsf{true}, \frac{1}{2}\mathsf{false}\}$
return $\mathcal{M}$

$\mathcal{A}$.commit(c)

$r \leftarrow\$$ Rand
$(\hat{c}, \hat{d}) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(\mathsf{false}, \mathsf{r})$
$R \leftarrow \lambda m_0 m_1.(m_0 = \mathsf{false}) \wedge\ (m_1 = \mathsf{false})$
return $(1, R, \hat{c})$

$\mathcal{A}$.decommit(d)

if $\mathsf{Verify}_{\mathsf{pk}}(\mathsf{false}, \mathsf{c}, \mathsf{d})$
return $(\mathsf{false}, \hat{d})$
else fail

COUNTER-EXAMPLE

- $\Pr\left[GN_1^{\mathcal{A}} : m = false \ \wedge \bar{m} = false \ \wedge win\right] \approx \frac{1}{4}$.
- $\Pr\left[GN_1^{\mathcal{A}} : m = false \wedge \bar{m} = true \wedge win\right] = 0.$
- $\Pr\left[GN_1^{\mathcal{A}} : m = true \wedge \bar{m} = false \wedge win\right] = 0.$
- $\Pr\left[GN_1^{\mathcal{A}} : m = true \wedge \bar{m} = true \wedge win\right] = 0.$

$GN_1^{\mathcal{A}}$

1 : $\mathsf{pk} \leftarrow \mathsf{Gen}$
2 : $r \leftarrow\$ \mathsf{Rand}$
3 : $\mathcal{M} \leftarrow \mathcal{A}.\mathsf{init}(\mathsf{pk})$
4 : $m \leftarrow\$ \mathcal{M}; \bar{m} \leftarrow\$ \mathcal{M}$
5 : $(\bar{c}, \bar{d}) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(\bar{m}, r)$
6 : $(n, R(\cdot), \hat{c}_1, ..., \hat{c}_n) \leftarrow \mathcal{A}.\mathsf{commit}(\bar{c})$
7 : $((\hat{m}_1, \hat{d}_1), ..., (\hat{m}_n, \hat{d}_n)) \leftarrow \mathcal{A}.\mathsf{decommit}(\bar{d})$
8 : if $\bar{c} \in \{\hat{c}_1, ..., \hat{c}_n\}$ **return** 0
9 : $b \leftarrow \mathsf{Verify}_{\mathsf{pk}}(\hat{m}_i, \hat{c}_i, \hat{d}_i)\ \forall i \in \{1 \leq i \leq n\}$
10 : **return** $R(m, \hat{m}_1, ..., \hat{m}_n)$

$\mathcal{A}.\mathsf{init}(\mathrm{pk})$

$\mathcal{M} \leftarrow \{\frac{1}{2}\mathsf{true}, \frac{1}{2}\mathsf{false}\}$
return $\mathcal{M}$

$\mathcal{A}.\mathsf{commit}(\mathrm{c})$

$r \leftarrow\$ \mathsf{Rand}$
$(\hat{c}, \hat{d}) \leftarrow \mathsf{Commit}_{\mathsf{pk}}(\mathsf{false}, \mathsf{r})$
$R \leftarrow \lambda m_0 m_1.(m_0 = \mathsf{false}) \wedge\ (m_1 = \mathsf{false})$
return $(1, R, \hat{c})$

$\mathcal{A}.\mathsf{decommit}(\mathrm{d})$

if $\mathsf{Verify}_{\mathsf{pk}}(\mathsf{false}, \mathsf{c}, \mathsf{d})$
return $(\mathsf{false}, \hat{d})$
else fail

COUNTER-EXAMPLE

- Adversary can distinguish these two games with the following probability:

$$\mathrm{Adv}_{\mathcal{C}}^{\mathrm{nmo}}(\mathcal{A}) = \left|\Pr\left[GN_0^{\mathcal{A}} = 1\right] - \Pr\left[GN_1^{\mathcal{A}} = 1\right]\right| = \frac{1}{4}.$$

CONCLUSION

- Comparison-based non-malleability (Laur and Nyberg, 2007) is unsatisfiable.
- Pen-and-paper proofs are unreliable.
- Formal methods are of great importance for cryptography.
- Thank you ☺.