(Un)satisfiability of Comparison-Based Non-Malleability for Commitments

Denis Firsov, Sven Laur, and Ekaterina Zhuchko

Motivation

- Problem: developing cryptographic proofs is a tedious and error-prone task.

"In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor."

(Bellare and Rogeway, 2004)

"We generate more proofs than we carefully verify."

(Hallevi, 2005)

"Security proofs for even simple cryptographic systems are dangerous and ugly beasts."

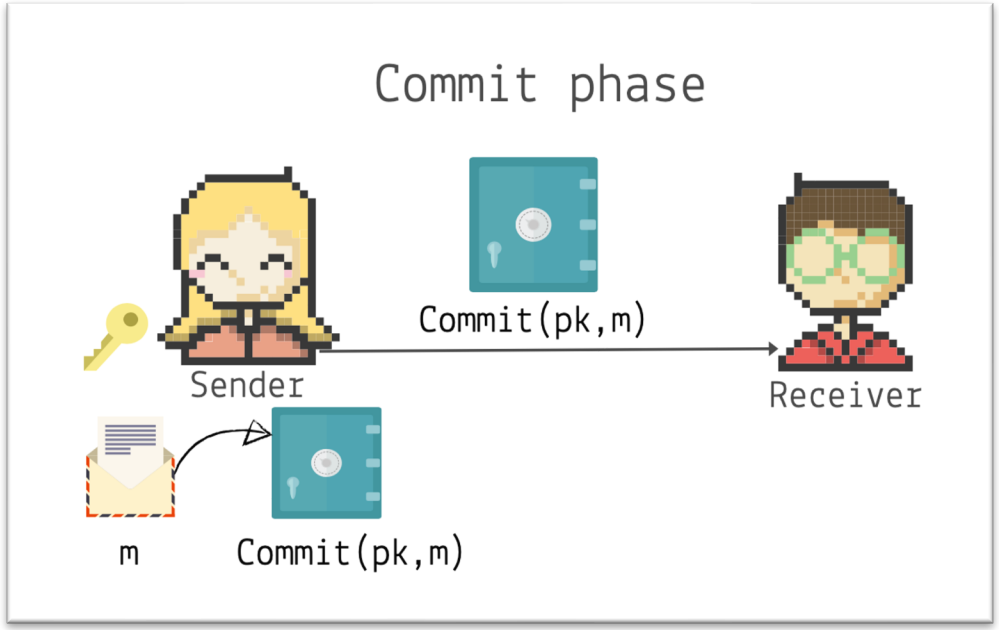
(Bristol Crypto Group, 2017)

Motivation

- Formal methods to the rescue!
- Pen-and-paper proofs vs. formally verified proofs (machine-checked).
- EasyCrypt is an interactive theorem prover for reasoning about cryptographic schemes and definitions.

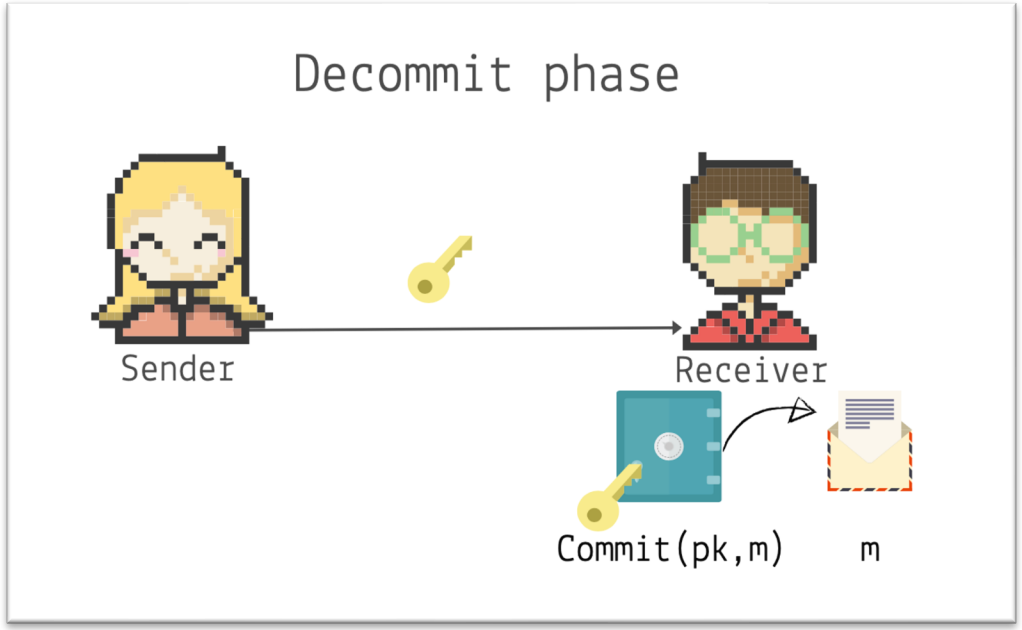
Commitment Schemes

- Two participating parties: sender and receiver.
- Two-phase protocol: commit and decommit.
- The sender has a private message.
- The sender commits to a message and sends it to the receiver.

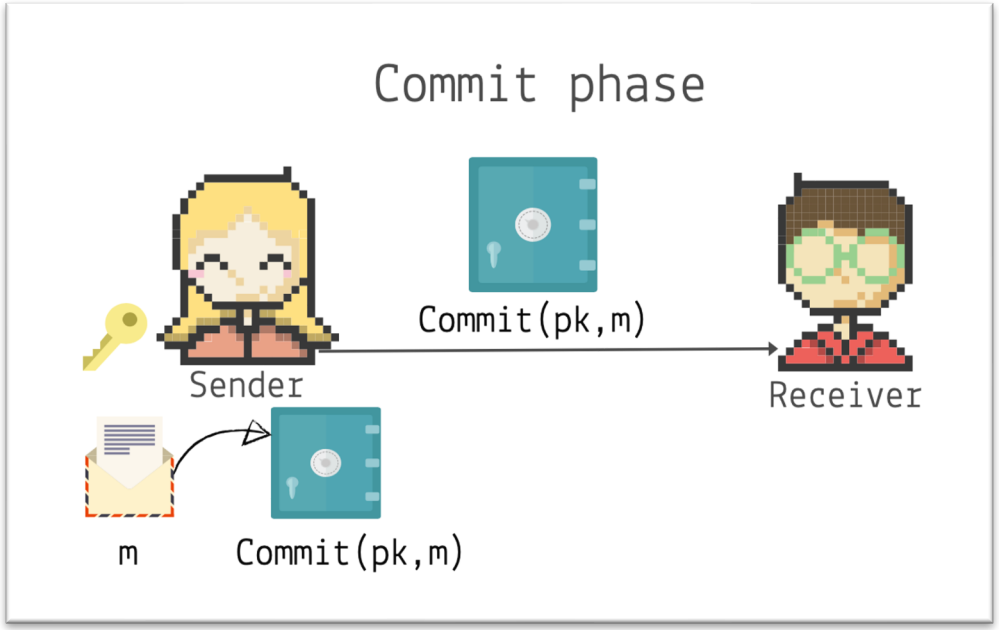


Commitment Schemes

- Two participating parties: sender and receiver.
- Two-phase protocol: commit and decommit.
- The sender has a private message.
- The sender commits to a message and sends it to the receiver.
- At a later stage, the receiver can open the commitment and read the message.

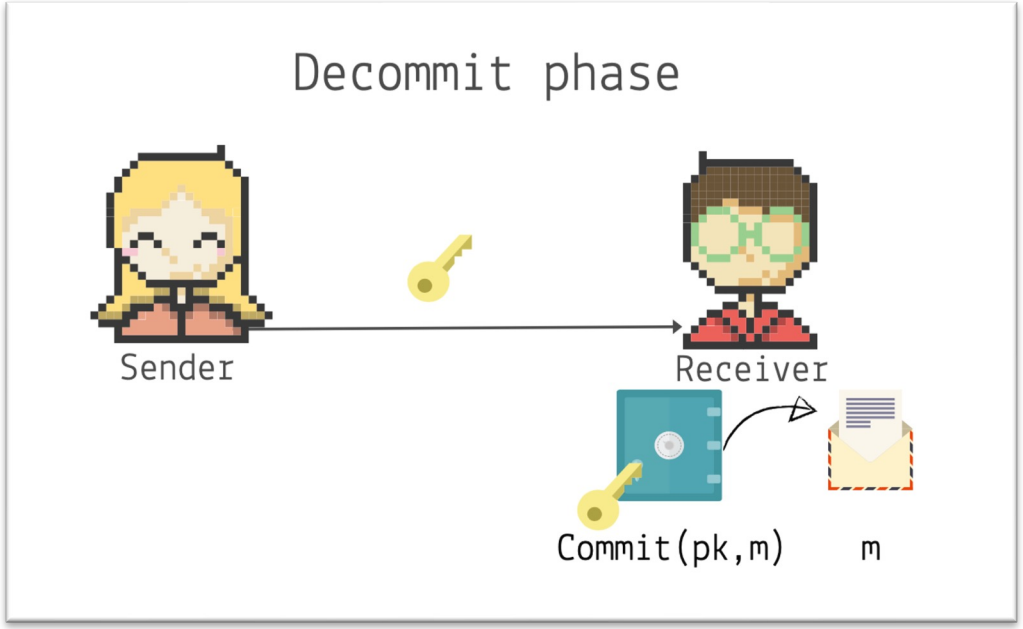


Security Properties: Hiding



- Hiding: the receiver cannot see the message inside the commitment.
- Note that $Commit(pk, m)$ is a parameterised distribution of commitments.

Security Properties: Binding



- Binding: the sender is bound to the committed message.
- The commitment should not have any secret backdoors i.e. open to two different messages.

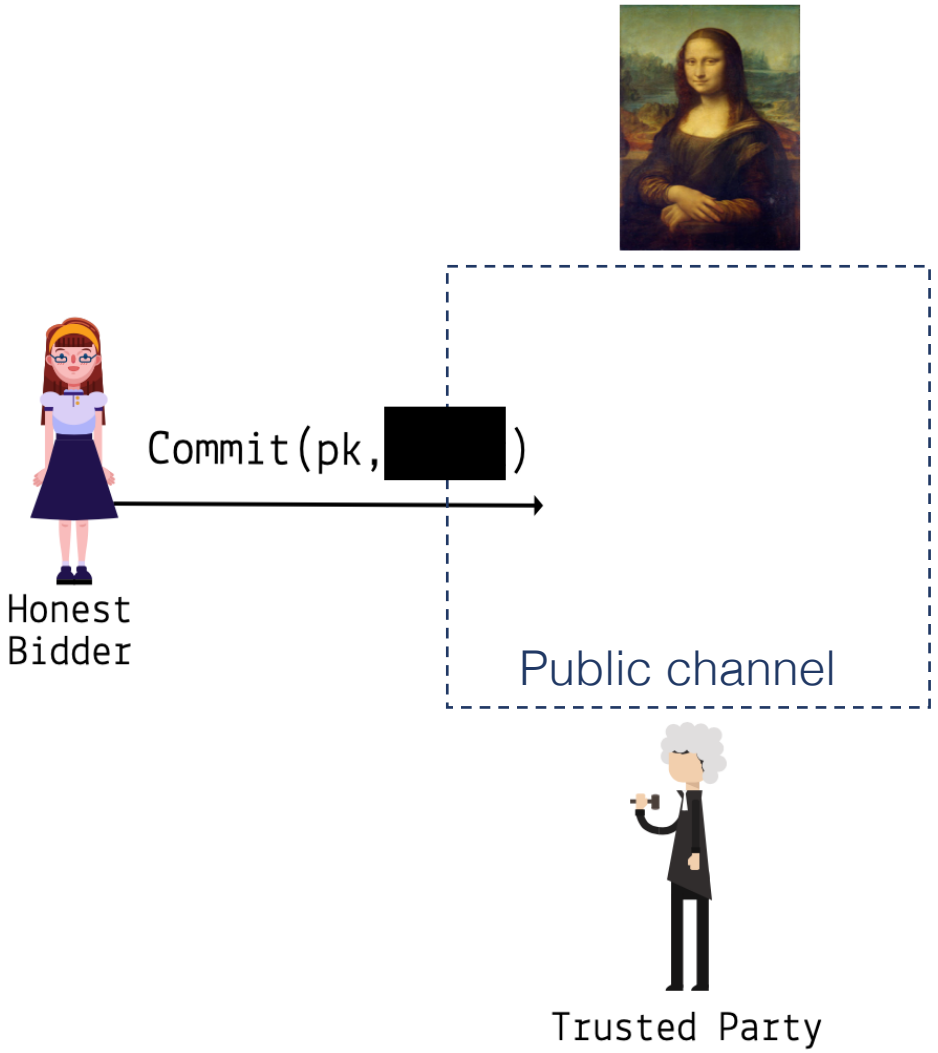
Motivating Example: Blind Auction

Trusted Party

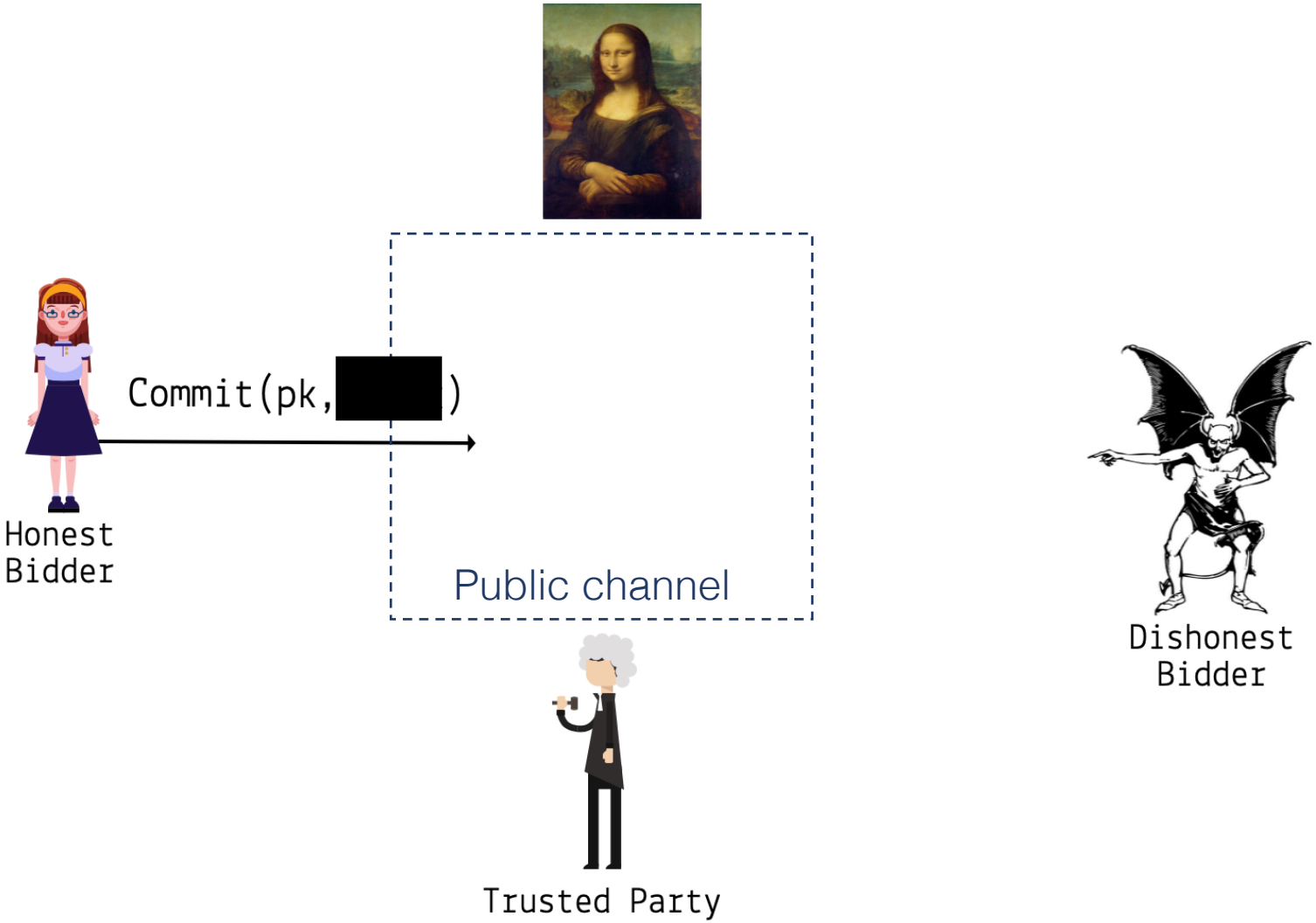
Motivating Example: Blind Auction

Trusted Party

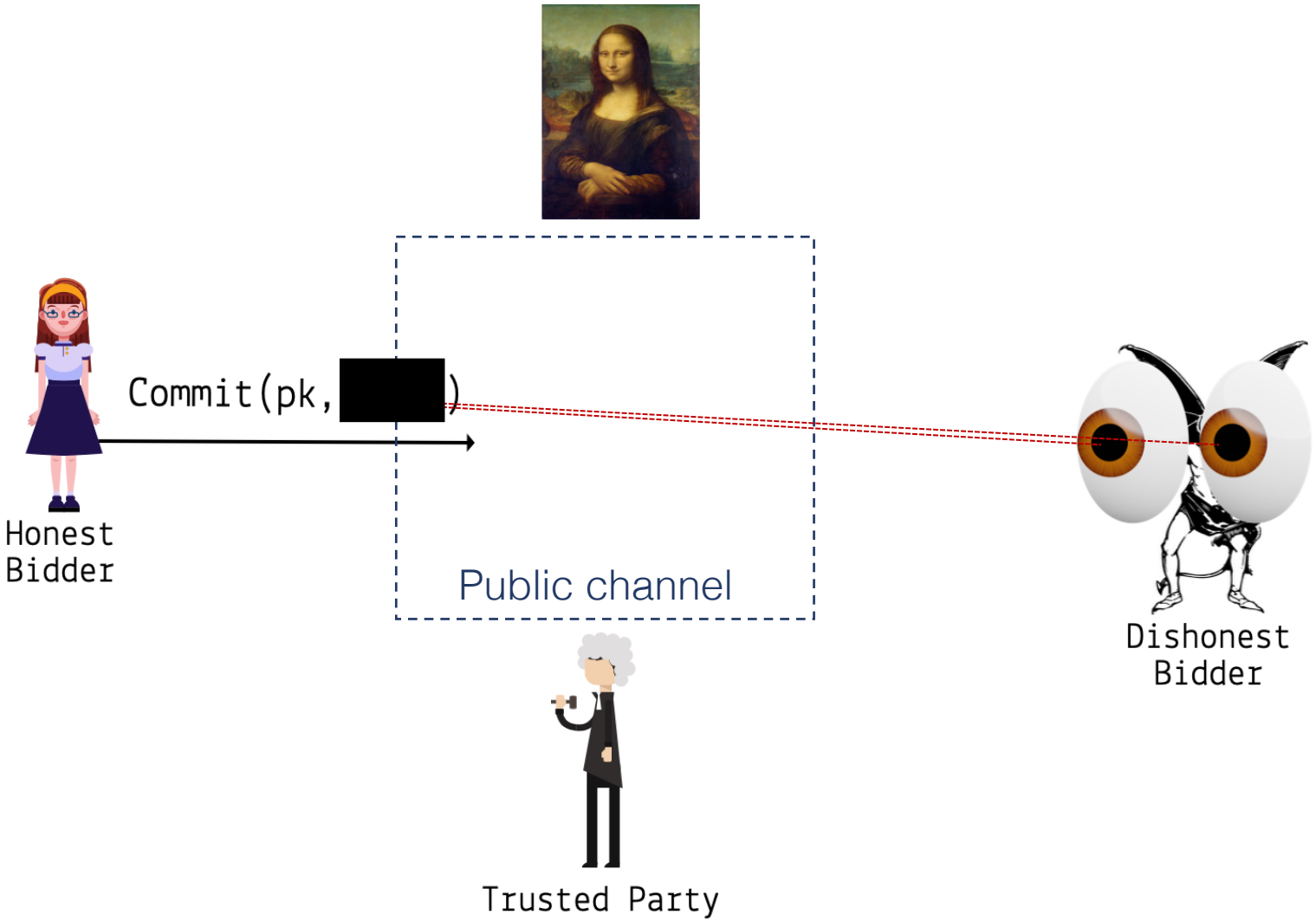
Motivating Example: Blind Auction



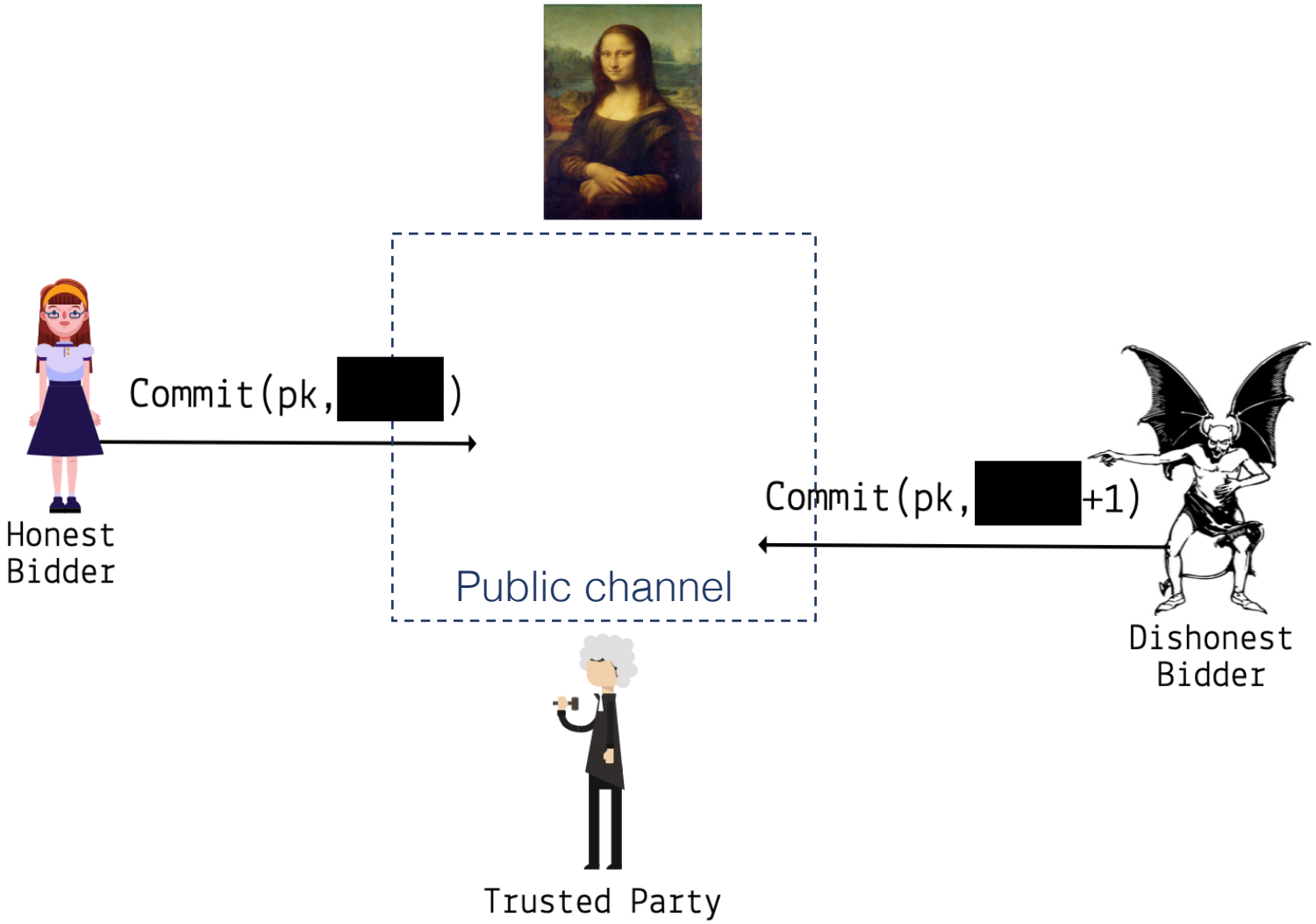
Motivating Example: Blind Auction



Motivating Example: Blind Auction



Motivating Example: Blind Auction



Motivating Example: Blind Auction

- We have a commitment scheme that is both hiding and binding.
- The attack scenario is possible due to the malleability of the commitment scheme.

Contribution

- There are two ways to define non-malleability:
  - Simulation-based definition;
  - Comparison-based definition.

Contribution

- There are two ways to define non-malleability:
  - Simulation-based definition;
  - Comparison-based definition.
- Simulation-based definition is hard to falsify.
- Example of falsifiability:
  - Find collisions in SHA-256;
  - $SHA256(message_1) = hash = SHA256(message_2)$.

Contribution

- There are two ways to define non-malleability:
  - Simulation-based definition;
  - Comparison-based definition.
- Simulation-based definition is hard to falsify.
  - Our goal was to give a more falsifiable definition.

Contribution

- There are two ways to define non-malleability:
  - Simulation-based definition;
  - Comparison-based definition.
- Simulation-based definition is hard to falsify.
  - Our goal was to give a more falsifiable definition.
- But we show that:
  - It is unsatisfiable for all “good” commitment schemes.
  - It is satisfiable for the “bad” commitment schemes.

Comparison-Based Non-Malleability

A commitment scheme $C = (Gen, Commit, Verify)$ *is* ***comparison-based non-malleable*** *iff for any efficient adversary* A*, the advantage* $AdvC(C, A)$ *is negligible, where*

$$AdvC(C, A) := |\mathsf{Pr}\left[r \leftarrow GN_0(C, A).main() : r = 1\right] - \mathsf{Pr}\left[r \leftarrow GN_1(C, A).main() : r = 1\right]|.$$

$GN_0(C, A)$	$GN_1(C, A)$
$pk \xleftarrow{\$} Gen$	$pk \xleftarrow{\$} Gen$
$\mathcal{M} \leftarrow A.init(pk)$	$\mathcal{M} \leftarrow A.init(pk)$
$m \xleftarrow{\$} \mathcal{M}$	$m \xleftarrow{\$} \mathcal{M}; n \xleftarrow{\$} \mathcal{M}$
$(c, d) \xleftarrow{\$} Commit(pk, m)$	$(c, d) \xleftarrow{\$} Commit(pk, m)$
$(c', R) \leftarrow A.commit(c)$	$(c', R) \leftarrow A.commit(c)$
$(d', m') \leftarrow A.decommit(d)$	$(d', m') \leftarrow A.decommit(d)$
$v \leftarrow Verify(pk, m', c', d')$	$v \leftarrow Verify(pk, m', c', d')$
return $v \wedge R(m, m') \wedge c \neq c'$	**return** $v \wedge R(n, m') \wedge c \neq c'$

Comparison-Based Non-Malleability

A commitment scheme $C = (Gen, Commit, Verify)$ *is* ***comparison-based non-malleable*** *iff for any efficient adversary* A*, the advantage* $AdvC(C, A)$ *is negligible, where*

$$AdvC(C, A) := |\mathsf{Pr}\left[r \leftarrow GN_0(C, A).main() : r = 1\right] - \mathsf{Pr}\left[r \leftarrow GN_1(C, A).main() : r = 1\right]|.$$

$GN_0(C, A)$	$GN_1(C, A)$
$pk \xleftarrow{\$} Gen$	$pk \xleftarrow{\$} Gen$
$\mathcal{M} \leftarrow A.init(pk)$	$\mathcal{M} \leftarrow A.init(pk)$
$m \xleftarrow{\$} \mathcal{M}$	$m \xleftarrow{\$} \mathcal{M}; n \xleftarrow{\$} \mathcal{M}$
$(c, d) \xleftarrow{\$} Commit(pk, m)$	$(c, d) \xleftarrow{\$} Commit(pk, m)$
$(c', R) \leftarrow A.commit(c)$	$(c', R) \leftarrow A.commit(c)$
$(d', m') \leftarrow A.decommit(d)$	$(d', m') \leftarrow A.decommit(d)$
$v \leftarrow Verify(pk, m', c', d')$	$v \leftarrow Verify(pk, m', c', d')$
return $v \wedge R(m, m') \wedge c \neq c'$	**return** $v \wedge R(n, m') \wedge c \neq c'$

Comparison-Based Non-Malleability

A commitment scheme $C = (Gen, Commit, Verify)$ *is* ***comparison-based non-malleable*** *iff for any efficient adversary* A*, the advantage* $AdvC(C, A)$ *is negligible, where*

$$AdvC(C, A) := |\mathsf{Pr}\left[r \leftarrow GN_0(C, A).main() : r = 1\right] - \mathsf{Pr}\left[r \leftarrow GN_1(C, A).main() : r = 1\right]|.$$

$GN_0(C, A)$	$GN_1(C, A)$
1 : $pk \xleftarrow{\$} Gen$	1 : $pk \xleftarrow{\$} Gen$
2 : $\mathcal{M} \leftarrow A.init(pk)$	2 : $\mathcal{M} \leftarrow A.init(pk)$
3 : $m \xleftarrow{\$} \mathcal{M}$	3 : $m \xleftarrow{\$} \mathcal{M}; n \xleftarrow{\$} \mathcal{M}$
4 : $(c, d) \xleftarrow{\$} Commit(pk, m)$	4 : $(c, d) \xleftarrow{\$} Commit(pk, m)$
5 : $(c', R) \leftarrow A.commit(c)$	5 : $(c', R) \leftarrow A.commit(c)$
6 : $(d', m') \leftarrow A.decommit(d)$	6 : $(d', m') \leftarrow A.decommit(d)$
7 : $v \leftarrow Verify(pk, m', c', d')$	7 : $v \leftarrow Verify(pk, m', c', d')$
8 : **return** $v \wedge R(m, m') \wedge c \neq c'$	8 : **return** $v \wedge R(n, m') \wedge c \neq c'$

Comparison-Based Non-Malleability

A commitment scheme $C = (Gen, Commit, Verify)$ *is* ***comparison-based non-malleable*** *iff for any efficient adversary* A*, the advantage* $AdvC(C, A)$ *is negligible, where*

$$AdvC(C, A) := |\mathsf{Pr}\left[r \leftarrow GN_0(C, A).main() : r = 1\right] - \mathsf{Pr}\left[r \leftarrow GN_1(C, A).main() : r = 1\right]|.$$

$GN_0(C, A)$	$GN_1(C, A)$
$pk \xleftarrow{\$} Gen$	$pk \xleftarrow{\$} Gen$
$\mathcal{M} \leftarrow A.init(pk)$	$\mathcal{M} \leftarrow A.init(pk)$
$m \xleftarrow{\$} \mathcal{M}$	$m \xleftarrow{\$} \mathcal{M}; n \xleftarrow{\$} \mathcal{M}$
$(c, d) \xleftarrow{\$} Commit(pk, m)$	$(c, d) \xleftarrow{\$} Commit(pk, m)$
$(c', R) \leftarrow A.commit(c)$	$(c', R) \leftarrow A.commit(c)$
$(d', m') \leftarrow A.decommit(d)$	$(d', m') \leftarrow A.decommit(d)$
$v \leftarrow Verify(pk, m', c', d')$	$v \leftarrow Verify(pk, m', c', d')$
return $v \wedge R(m, m') \wedge c \neq c'$	**return** $v \wedge R(n, m') \wedge c \neq c'$

Comparison-Based Non-Malleability

A commitment scheme $C = (Gen, Commit, Verify)$ *is* ***comparison-based non-malleable*** *iff for any efficient adversary* A*, the advantage* $AdvC(C, A)$ *is negligible, where*

$$AdvC(C, A) := |\mathsf{Pr}\left[r \leftarrow GN_0(C, A).main() : r = 1\right] - \mathsf{Pr}\left[r \leftarrow GN_1(C, A).main() : r = 1\right]|.$$

$GN_0(C, A)$	$GN_1(C, A)$
$pk \xleftarrow{\$} Gen$	$pk \xleftarrow{\$} Gen$
$\mathcal{M} \leftarrow A.init(pk)$	$\mathcal{M} \leftarrow A.init(pk)$
$m \xleftarrow{\$} \mathcal{M}$	$m \xleftarrow{\$} \mathcal{M}; n \xleftarrow{\$} \mathcal{M}$
$(c, d) \xleftarrow{\$} Commit(pk, m)$	$(c, d) \xleftarrow{\$} Commit(pk, m)$
$(c', R) \leftarrow A.commit(c)$	$(c', R) \leftarrow A.commit(c)$
$(d', m') \leftarrow A.decommit(d)$	$(d', m') \leftarrow A.decommit(d)$
$v \leftarrow Verify(pk, m', c', d')$	$v \leftarrow Verify(pk, m', c', d')$
return $v \wedge R(m, m') \wedge c \neq c'$	**return** $v \wedge R(n, m') \wedge c \neq c'$

Adversary for Comparison-Based Non-Malleability

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

$A.pk \leftarrow pk$

return $\{0, 1\}$

- We construct an adversary A who executes a successful attack.
- A outputs a uniform distribution on bits.

Adversary for Comparison-Based Non-Malleability

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

$A.pk \leftarrow pk$

return $\{0,1\}$

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

$A.c \leftarrow c$

$R \leftarrow \lambda m_0 m_1 . m_0 = 0 \wedge m_1 = 0$

$(c', d') \xleftarrow{\$} Commit(pk, 0)$

return (R, c')

- The relation $\mathbf{R}(m, m')$ only holds when both messages are $\mathbf{0}$.
- A outputs the relation $\mathbf{R}$ and the commitment on message $\mathbf{0}$.

Adversary for Comparison-Based Non-Malleability

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

$A.pk \leftarrow pk$

return $\{0,1\}$

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

$A.c \leftarrow c$

$R \leftarrow \lambda m_0 m_1 . m_0 = 0 \wedge m_1 = 0$

$(c', d') \xleftarrow{\$} Commit(pk, 0)$

return (R, c')

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

if $Verify(pk, 0, c, d)$ **then**

  return $(d', 0)$

else $\perp$

- A checks if c was a commitment on the message $m = 0$:
  - $m = 0$, then return $(d', 0)$.
  - $m \neq 0$, then fail the game.

Adversary for Comparison-Based Non-Malleability

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

$A.pk \leftarrow pk$

return $\{0,1\}$

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

$A.c \leftarrow c$

$R \leftarrow \lambda m_0 m_1 . m_0 = 0 \wedge m_1 = 0$

$(c', d') \xleftarrow{\$} Commit(pk, 0)$

return (R, c')

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

if $Verify(pk, 0, c, d)$ **then**

  return $(d', 0)$

else $\perp$

- Goal: to calculate the advantage of A in winning the non-malleability games.

Theorem

For any functional commitment scheme $C = (Gen, Commit, Verify)$ *the adversary* A *has the following comparison-based non-malleability advantage:*

$$AdvC(C, A) = \frac{1}{4} - \frac{1}{4} \cdot \Pr\left[\begin{array}{c} pk \xleftarrow{\$} Gen;\ (c, d) \xleftarrow{\$} Commit(pk, 0); \\ (c', d') \xleftarrow{\$} Commit(pk, 0) : c = c' \end{array} \right].$$

Theorem

For any functional commitment scheme $C = (Gen, Commit, Verify)$ *the adversary* A *has the following comparison-based non-malleability advantage:*

$$AdvC(C, A) = \frac{1}{4} - \frac{1}{4} \cdot \Pr\left[\begin{array}{c} pk \xleftarrow{\$} Gen;\ (c, d) \xleftarrow{\$} Commit(pk, 0); \\ (c', d') \xleftarrow{\$} Commit(pk, 0) : c = c' \end{array}\right].$$

- If the commitments generated by $\boldsymbol{Commit}$ have enough randomness then $\boldsymbol{AdvC(C, A) \approx \frac{1}{4}}$ and is not negligible.

Some Intuition

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

$A.pk \leftarrow pk$

return $\{0,1\}$

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

$A.c \leftarrow c$

$R \leftarrow \lambda m_0 m_1 . m_0 = 0 \wedge m_1 = 0$

$(c', d') \xleftarrow{\$} Commit(pk, 0)$

return (R, c')

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

if $Verify(pk, 0, c, d)$ **then**

  return $(d', 0)$

else $\perp$

- A is able to successfully execute the attack because it has all the information during decommit.
- A opens the commitment and aborts the game if necessary.

Satisfiability for the “Bad Case”

- Constant commitment scheme, where $\{*\}$ is a singleton set:

$$Gen := \{*\}$$
$$Commit(pk, m) := (*, m)$$
$$Verify(pk, m, c, d) := \textbf{\textit{if}}\ m = d\ \textbf{\textit{then}}\ 1\ \textbf{\textit{else}}\ 0$$

- For any message, the commitment is a constant value.
- It is hiding and non-binding.
- It is also non-malleable according to the comparison-based definition.

Conclusion

- Comparison-based definition:
  - Is unsatisfiable for all realistic commitment schemes;
  - Is satisfiable for a paradoxical commitment scheme.
- It does require a lot of time and effort to formally verify a cryptographic proof.
  - Example: comparison-based proof was 4 lines on paper but 600 lines of code.
  - A lot of manual effort.
- Future work: application in timestamping.

Thank You!