
On Generating the Initial Key in the Bounded-Storage Model

Stefan Dziembowski

www.dziembowski.net

Warsaw University



Ueli Maurer

crypto.ethz.ch/~ueli

ETH Zurich



Main idea

Instead of assuming that
the **computing** power of the adversary is
limited
we assume that her
memory is limited.

Plan

1. A short introduction to the Bounded Storage Model. ←
2. Our contributions.

Motivation

Common practice in cryptography:

If you need an encryption scheme then take AES (or IDEA, RSA, ...). You can be confident that your adversary cannot break it.

Motivation

Common practice in cryptography:

If you need an encryption scheme then take AES (or IDEA, RSA, ...). You can be confident that your adversary cannot break it.

Why?

Motivation

Common practice in cryptography:

If you need an encryption scheme then take AES (or IDEA, RSA, ...). You can be confident that your adversary cannot break it.

Why?

Because several people tried to break these ciphers and nobody succeeded.

Motivation

Common practice in cryptography:

If you need an encryption scheme then take AES (or IDEA, RSA, ...). You can be confident that your adversary cannot break it.

Why?

Because several people tried to break these ciphers and nobody succeeded.

Question:

Can we also be confident that nobody will break it **in the future**?

A problem with any (classical) cryptosystem

The following attack is possible:

1. The adversary **stores** the transcript of your entire communication.

A problem with any (classical) cryptosystem

The following attack is possible:

1. The adversary **stores** the transcript of your entire communication.
2. Later (maybe in 30 years) he **decrypts** it.

A problem with any (classical) cryptosystem

The following attack is possible:

1. The adversary **stores** the transcript of your entire communication.
2. Later (maybe in 30 years) he **decrypts** it.

So, the secrecy is **not everlasting**.

A solution to the problem

How to solve this problem?

A solution to the problem

How to solve this problem?

We have to assume that the adversary cannot **store** the entire communication between the users.

A solution to the problem

How to solve this problem?

We have to assume that the adversary cannot **store** the entire communication between the users.

One of the following options come to mind:

1. make some non-standard assumptions about the communication channel (eg. quantum, noisy, ...)

A solution to the problem

How to solve this problem?

We have to assume that the adversary cannot **store** the entire communication between the users.

One of the following options come to mind:

1. make some non-standard assumptions about the communication channel (eg. quantum, noisy, ...)
2. simply assume that the amount of transferred data is too large to be stored in the **memory** of the adversary.

A solution to the problem

How to solve this problem?

We have to assume that the adversary cannot **store** the entire communication between the users.

One of the following options come to mind:

1. make some non-standard assumptions about the communication channel (eg. quantum, noisy, ...)
2. simply assume that the amount of transferred data is too large to be stored in the **memory** of the adversary.

We will follow option 2 — the **Bounded-Storage Model (BSM)**.

More on the model

We will construct protocols such that:

The transcript of their execution is too large to fit in *Eve's* memory.

More on the model

We will construct protocols such that:

The transcript of their execution is too large to fit in *Eve's* memory.

To make it more practical we will assume that:

There is no need for the users to transmit large amounts of data.

More on the model

We will construct protocols such that:

The transcript of their execution is too large to fit in *Eve's* memory.

To make it more practical we will assume that:

There is no need for the users to transmit large amounts of data.

We will assume that the parties have an access to a **very large** string of bits called a **randomizer**.

The randomizer

The randomizer

- may be broadcast by a satellite,

The randomizer

The randomizer

- may be broadcast by a satellite, or
- may come from an astronomical observation.
- may be generated by one of the users of the protocol

The randomizer

The randomizer

- may be broadcast by a satellite, or
- may come from an astronomical observation.
- may be generated by one of the users of the protocol

It is a very long string of random bits

- The users of the protocol need to access only a small part of it.

The randomizer

The randomizer

- may be broadcast by a satellite, or
- may come from an astronomical observation.
- may be generated by one of the users of the protocol

It is a very long string of random bits

- The users of the protocol need to access only a small part of it.
- *Eve* cannot store the entire randomizer in her memory.

Nice fact about the BSM

Let us assume that

the memory of *Eve* is smaller than the length of the randomizer.

(a precise bound on *Eve's* memory depends on the setting).

Nice fact about the BSM

Let us assume that

the memory of *Eve* is smaller than the length of the randomizer.

(a precise bound on *Eve's* memory depends on the setting).

It turns out that:

We can do **provably secure** cryptography
(**encryption, oblivious transfer, key-agreement**)
without any further assumptions!

This is in contrast with the standard (complexity-based) cryptography.

Secret-key encryption in the BSM

Secret-key encryption schemes in the BSM can be viewed as

stream-ciphers

Secret-key encryption in the BSM

Secret-key encryption schemes in the BSM can be viewed as

stream-ciphers

Honest users *Alice* (A) and *Bob* (B)

sharing a short **initial key** K

Secret-key encryption in the BSM

Secret-key encryption schemes in the BSM can be viewed as

stream-ciphers

Honest users *Alice* (A) and *Bob* (B)

sharing a short **initial key** K

want to

securely derive a longer **derived key** X .

Secret-key encryption in the BSM

Secret-key encryption schemes in the BSM can be viewed as

stream-ciphers

Honest users *Alice* (A) and *Bob* (B)

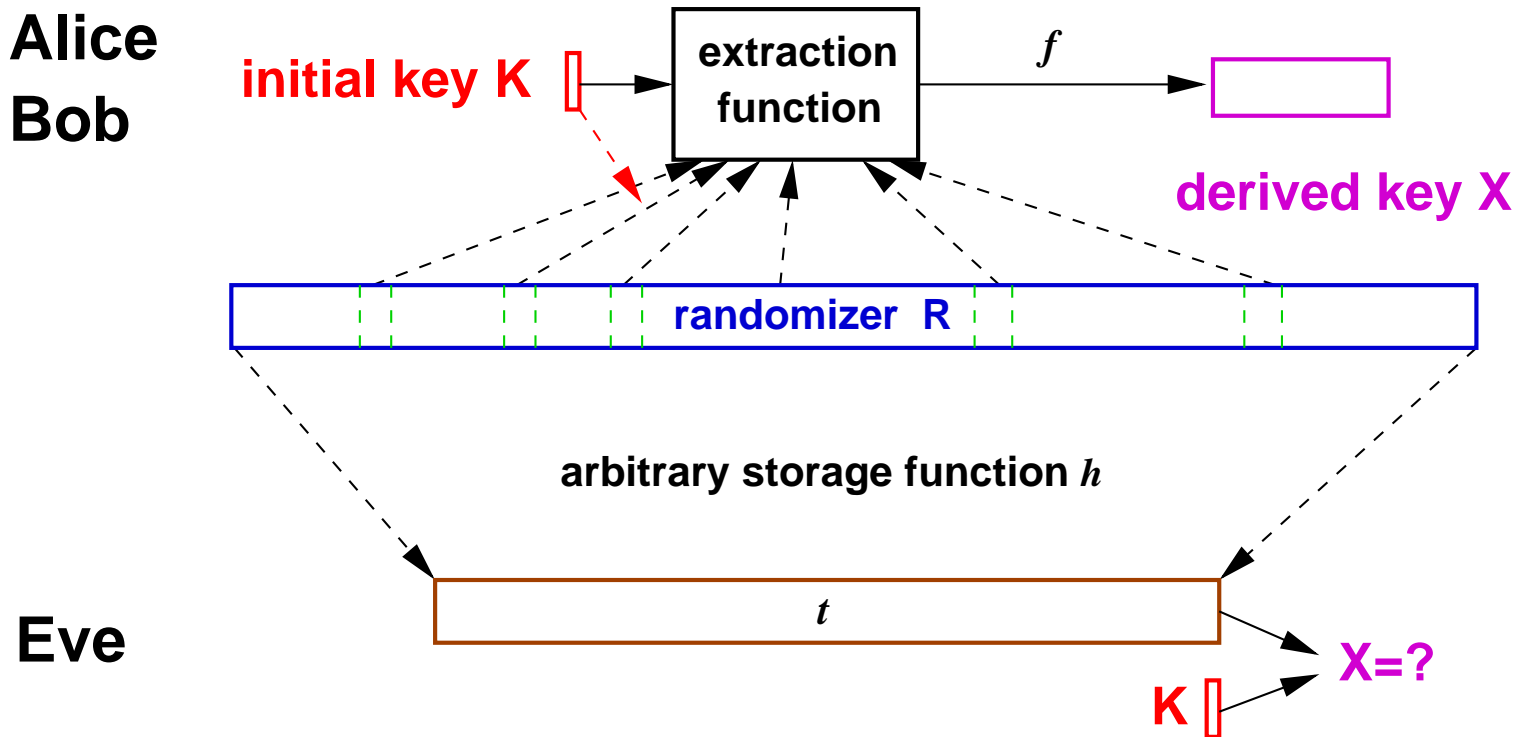
sharing a short **initial key** K

want to

securely derive a longer **derived key** X .

X can be used e.g. as a **one-time pad** for a secure encryption or for **authentication**.

Secret-key derivation



Previous results

Main task:

Find functions f that are secure for as liberal as possible memory bound

Several functions were proven secure during the last decade [Maurer92, CachinMaurer97, AumannRabin99, AumannDingRabin02, DziembowskiMaurer02, Lu02, Vadhan03, ...].

The scheme of Aumann and Rabin

The simplest one is a function deriving **one bit**:

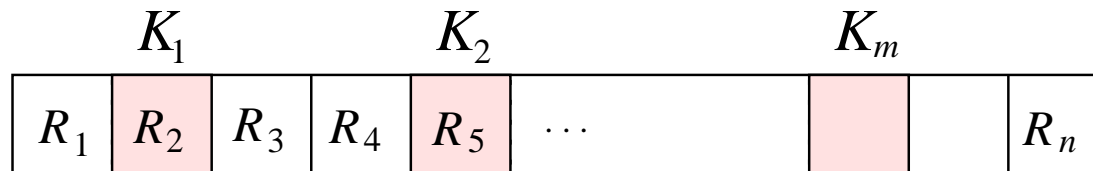
The scheme of Aumann and Rabin

The simplest one is a function deriving **one bit**:

m — security parameter

n — length of the randomizer

initial key: $K = (K_1, \dots, K_m)$, with $1 \leq K_1 < \dots < K_m \leq n$.



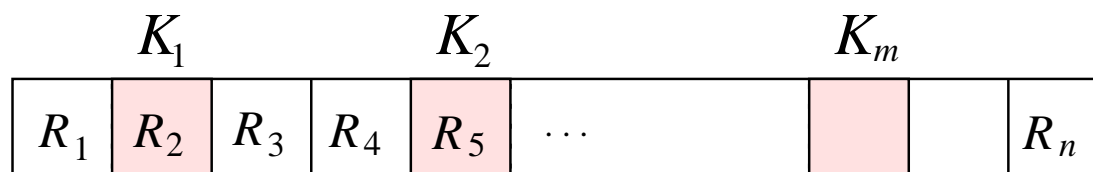
The scheme of Aumann and Rabin

The simplest one is a function deriving **one bit**:

m — security parameter

n — length of the randomizer

initial key: $K = (K_1, \dots, K_m)$, with $1 \leq K_1 < \dots < K_m \leq n$.



For a randomizer $R = (R_1, \dots, R_n)$ the value of the derived key is equal to

$$f(K, R) = R_{K_1} \oplus \dots \oplus R_{K_m} \in \{0, 1\}$$

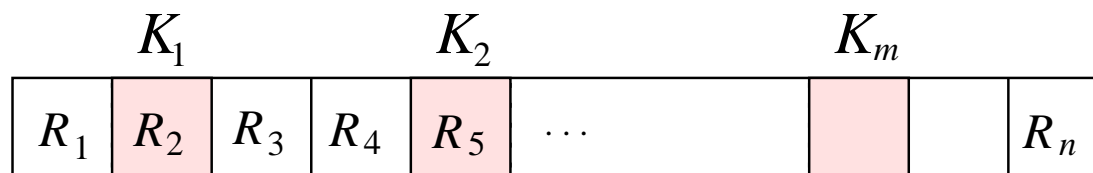
The scheme of Aumann and Rabin

The simplest one is a function deriving **one bit**:

m — security parameter

n — length of the randomizer

initial key: $K = (K_1, \dots, K_m)$, with $1 \leq K_1 < \dots < K_m \leq n$.



For a randomizer $R = (R_1, \dots, R_n)$ the value of the derived key is equal to

$$f(K, R) = R_{K_1} \oplus \dots \oplus R_{K_m} \in \{0, 1\}$$

This function was proven secure if the **Eve's** memory is at most $n/5$.

Plan

1. A short introduction to the Bounded Storage Model. ✓
2. Our contributions. ←

Q: How to generate the initial key?

The initial key can be generated:

- in the BSM itself.

this is called a **secret-key agreement in the BSM.**

Q: How to generate the initial key?

The initial key can be generated:

- in the BSM itself.

this is called a **secret-key agreement in the BSM**.

- using classical (complexity-based) methods.

We call it a **hybrid-model**.

We will discuss the details in a moment.

Plan

1. A short introduction to the Bounded Storage Model. ✓
2. Our contributions.
 - Key-Agreement in the BSM ←
 - Hybrid Model

Key agreement in the BSM

The scenario for the **key agreement in the BSM** is essentially the same as for the secret key-derivation, with the following differences:

Key agreement in the BSM

The scenario for the **key agreement in the BSM** is essentially the same as for the secret key-derivation, with the following differences:

- *Alice* and *Bob* don't share any initial key.

Key agreement in the BSM

The scenario for the **key agreement in the BSM** is essentially the same as for the secret key-derivation, with the following differences:

- *Alice* and *Bob* don't share any initial key.
- It's essential that the algorithms for *Alice* and *Bob* are **randomized**.

Key agreement in the BSM

The scenario for the **key agreement in the BSM** is essentially the same as for the secret key-derivation, with the following differences:

- *Alice* and *Bob* don't share any initial key.
- It's essential that the algorithms for *Alice* and *Bob* are **randomized**.

It was already studied in

C. Cachin and **U. Maurer** **Unconditional security against memory-bounded adversaries.** *CRYPTO* 1997.

Our result

a — the memory size of *Alice*

b — the memory size of *Bob*

Our result

a — the memory size of *Alice*

b — the memory size of *Bob*

t — the memory size of *Eve*

Our result

a — the memory size of *Alice*

b — the memory size of *Bob*

t — the memory size of *Eve*

X — agreed key

Our result

a — the memory size of *Alice*

X — agreed key

b — the memory size of *Bob*

t — the memory size of *Eve*

Theorem.

$$H(X) \leq \frac{ab}{t}$$

Our result

a — the memory size of *Alice*

X — agreed key

b — the memory size of *Bob*

t — the memory size of *Eve*

Theorem.

$$H(X) \leq \frac{ab}{t}$$

Note that it also implies a similar bound for the Oblivious Transfer protocols in the BSM.

Plan

1. A short introduction to the Bounded Storage Model. ✓
2. Our contributions.
 - Key-Agreement in the BSM ✓
 - Hybrid Model ←

The hybrid model (1/2)

In the **KA-hybrid model** the initial key is generated by classical (complexity-based) method **KA**.

The hybrid model (1/2)

In the **KA-hybrid model** the initial key is generated by classical (complexity-based) method **KA**.

Clearly, in this model we cannot assume that **Eve** is computationally unbounded

as long as the randomizer is available

The hybrid model (1/2)

In the **KA-hybrid model** the initial key is generated by classical (complexity-based) method **KA**.

Clearly, in this model we cannot assume that **Eve** is computationally unbounded

as long as the randomizer is available

However it seems that

later

we can give her an infinite computing power.

The hybrid model (2/2)

The „hybrid model” was informally argued to be secure in the following sense:

The derived key is secret (unless *Eve* breaks the „classical” key agreement before the randomizer disappears).

The hybrid model (2/2)

The „hybrid model” was informally argued to be secure in the following sense:

The derived key is secret (unless *Eve* breaks the „classical” key agreement before the randomizer disappears).

So, the security is still everlasting!

The hybrid model (2/2)

The „hybrid model” was informally argued to be secure in the following sense:

The derived key is secret (unless *Eve* breaks the „classical” key agreement before the randomizer disappears).

We show that this reasoning is **false!**

Our main observation

We show the following:

Theorem. For any function f that is **secure** in the BSM

Our main observation

We show the following:

Theorem. For any function f that is **secure** in the BSM there exists a computationally **secure** key-agreement scheme KA

Our main observation

We show the following:

Theorem. For any function f that is **secure** in the BSM there exists a computationally **secure** key-agreement scheme KA such that f is completely **insecure** in the KA -hybrid model.

Our main observation

We show the following:

Theorem. For any function f that is **secure** in the BSM there exists a computationally **secure** key-agreement scheme KA such that f is completely **insecure** in the KA -hybrid model.

KA is very artificial.

Our main observation

We show the following:

Theorem. For any function f that is **secure** in the BSM there exists a computationally **secure** key-agreement scheme KA such that f is completely **insecure** in the KA -hybrid model.

KA is very artificial.

For simplicity we assume that f is the one-bit function of [AumannRabin99].

Private Information Retrieval (1/2)

PIR is a protocol between two parties:

Private Information Retrieval (1/2)

PIR is a protocol between two parties:

- A user U holding an input $i \in \{1, \dots, n\}$.

Private Information Retrieval (1/2)

PIR is a protocol between two parties:

- A user U holding an input $i \in \{1, \dots, n\}$.
- A database D holding an input $V = (V_1, \dots, V_n) \in \{0, 1\}^n$

Private Information Retrieval (1/2)

PIR is a protocol between two parties:

- A user U holding an input $i \in \{1, \dots, n\}$.
- A database D holding an input $V = (V_1, \dots, V_n) \in \{0, 1\}^n$

At the end of the protocol:

Private Information Retrieval (1/2)

PIR is a protocol between two parties:

- A user U holding an input $i \in \{1, \dots, n\}$.
- A database D holding an input $V = (V_1, \dots, V_n) \in \{0, 1\}^n$

At the end of the protocol:

- U should output V_i .

Private Information Retrieval (1/2)

PIR is a protocol between two parties:

- A user U holding an input $i \in \{1, \dots, n\}$.
- A database D holding an input $V = (V_1, \dots, V_n) \in \{0, 1\}^n$

At the end of the protocol:

- U should output V_i .
- D should not learn any information about i .

Private Information Retrieval (2/2)

Every PIR protocol should satisfy the following:

The total number of bits exchanged between the parties has to be much smaller than the the length of V .

Private Information Retrieval (2/2)

Every PIR protocol should satisfy the following:

The total number of bits exchanged between the parties has to be much smaller than the the length of V .

A typical PIR protocol consists of 3 rounds:

1. The user sends a **query** $Q(i)$ to the database.
2. The database sends a **reply** $S := \mathcal{S}(Q(i), V)$.
3. The user computes an **answer** $\mathcal{A}(S, i)$.

Private Information Retrieval (2/2)

Every PIR protocol should satisfy the following:

The total number of bits exchanged between the parties has to be much smaller than the the length of V .

A typical PIR protocol consists of 3 rounds:

1. The user sends a **query** $Q(i)$ to the database.
2. The database sends a **reply** $S := \mathcal{S}(Q(i), V)$.
3. The user computes an **answer** $\mathcal{A}(S, i)$.

For example, the historically first (single-server) PIR scheme of **Kushilevitz** and **Ostrovsky** has this form.

The construction of KA

DH — the Diffie-Hellman protocol

PIR — the protocol of [KO97].

The construction of KA

DH — the Diffie-Hellman protocol

PIR — the protocol of [KO97].

The new key-agreement scheme KA :

The construction of KA

DH — the Diffie-Hellman protocol

PIR — the protocol of [KO97].

The new key-agreement scheme KA:

1. *Alice* and *Bob* invoke DH. Let $K = (K_1, \dots, K_m)$ be the agreed key.

The construction of KA

DH — the Diffie-Hellman protocol

PIR — the protocol of [KO97].

The new key-agreement scheme KA:

1. *Alice* and *Bob* invoke DH. Let $K = (K_1, \dots, K_m)$ be the agreed key.
2. *Alice* sends to *Bob* the PIR queries: $Q(K_1), \dots, Q(K_m)$.

The construction of KA

DH — the Diffie-Hellman protocol

PIR — the protocol of [KO97].

The new key-agreement scheme KA:

1. *Alice* and *Bob* invoke DH. Let $K = (K_1, \dots, K_m)$ be the agreed key.
2. *Alice* sends to *Bob* the PIR queries: $Q(K_1), \dots, Q(K_m)$.
3. The agreed key is K .

The construction of KA

DH — the Diffie-Hellman protocol

PIR — the protocol of [KO97].

The new key-agreement scheme KA:

1. *Alice* and *Bob* invoke DH. Let $K = (K_1, \dots, K_m)$ be the agreed key.
2. *Alice* sends to *Bob* the PIR queries: $Q(K_1), \dots, Q(K_m)$.
3. The agreed key is K .

(The only reason for having Step 2 is to include $Q(K_1), \dots, Q(K_m)$ in the transcript of KA.)

KA is secure

We now have the following:

Claim: Assuming PIR and DH are computationally secure the key agreement KA is computationally secure.

KA is secure

We now have the following:

Claim: Assuming PIR and DH are computationally secure the key agreement KA is computationally secure.

The proof is standard (we skip it here).

The attack

1. In the first phase:

- For each $Q(K_i)$ *Eve* acts as a database (with contents R) and computes $S_i := \mathcal{S}(Q(K_i), R)$. She stores these values.

The attack

1. In the first phase:

- For each $Q(K_i)$ *Eve* acts as a database (with contents R) and computes $S_i := \mathcal{S}(Q(K_i), R)$. She stores these values.
- She also stores the transcript T of the DH-key-agreement.

The attack

1. In the first phase:
 - For each $Q(K_i)$ *Eve* acts as a database (with contents R) and computes $S_i := \mathcal{S}(Q(K_i), R)$. She stores these values.
 - She also stores the transcript T of the DH-key-agreement.
2. Later (when *Eve* gets infinite computing power):
 - (a) She computes $K = (K_1, \dots, K_m)$ from T .

The attack

1. In the first phase:
 - For each $Q(K_i)$ *Eve* acts as a database (with contents R) and computes $S_i := \mathcal{S}(Q(K_i), R)$. She stores these values.
 - She also stores the transcript T of the DH-key-agreement.
2. Later (when *Eve* gets infinite computing power):
 - (a) She computes $K = (K_1, \dots, K_m)$ from T .
 - (b) She computes each $R_{K_i} = \mathcal{A}(S_i, K_i)$.

The attack

1. In the first phase:
 - For each $Q(K_i)$ *Eve* acts as a database (with contents R) and computes $S_i := \mathcal{S}(Q(K_i), R)$. She stores these values.
 - She also stores the transcript T of the DH-key-agreement.
2. Later (when *Eve* gets infinite computing power):
 - (a) She computes $K = (K_1, \dots, K_m)$ from T .
 - (b) She computes each $R_{K_i} = \mathcal{A}(S_i, K_i)$.
So, she can compute the derived key!

Open problem

The key-agreement protocol in our example is very artificial.

One may conjecture that all „**natural**” key-agreement protocols are „safe” in the context of the BSM.

Open problem

The key-agreement protocol in our example is very artificial. One may conjecture that all „**natural**” key-agreement protocols are „safe” in the context of the BSM.

Question: How to formalize the property of being „**natural**”?