Verified Security of Hashed Authenticated Key-Exchange Protocols

Work in progress

G. Barthe[1] J. M. Crespo[1] C. Ene[2] C. Kunz[1]
Y. Lakhnech[2] B. Gregoire[3] S. Zanella Béguelin[4]

[1]IMDEA Software, Madrid, Spain
[2]University of Grenoble, CNRS - VERIMAG, France
[3]INRIA, Sophia-Antipolis
[4]MSR, Cambridge, UK

FCC 2012

Key Exchange Protocols

A pool of n participants try to establish a comon symmetric session key, in the presence of an adversary.

Security Goals

Definition (Key privacy)

No one except the legitimate parties can distinguish the established key from a random key.

Definition (implicit key authentication)

In presence of a passive adversary, two legitimate parties engaged in a session, will compute the same key.

Not to be confused with key confirmation: no one can make a legitimate partner believe he shares a key with a partner, unless this is the case.

Diffie-Hellman

$$
\begin{array}{ccc}
A & & B \\
x \xleftarrow{\$} \mathbb{Z}_q & \xrightarrow{\quad A,\, X = g^x \quad} & \\
 & \xleftarrow{\quad B,\, Y = g^y \quad} & y \xleftarrow{\$} \mathbb{Z}_q
\end{array}
$$

Hashed DH-based key Exchange Protocols

- Participants exchange messages that allow each party to compute at the end of a session a bitstring, called *session string*.
- The session key is derived from the session string by applying a hash function.

Two types of protocols :

- the flow of messages is that of DH; it is the way the session string is computed that guarantees authentication. May suppose public keys, or symmetric keys,... . Matsumoto, Takashima and Imai'86
- the flow of messages is modified modified, e.g., by adding signatures, encryptions, further rounds,.... Bellovin and Merrit'92

HMQV

A variant of MQV.

$$A = g^a,\ B = g^b$$

$$\begin{array}{ccc}
A & & B \\
x \xleftarrow{\$} \mathbb{Z}_q & \xrightarrow{\quad A,\, X = g^x \quad} & \\
 & \xleftarrow{\quad B,\, Y = g^y \quad} & y \xleftarrow{\$} \mathbb{Z}_q
\end{array}$$

$$\textsc{ses}(A, B, g^x, g^y) = g^{(x+H_1(B,X)a)(y+H_1(A,Y)b)}$$

$$K = H(\textsc{ses}(A, B, g^x, g^y))$$

Initializaton:
n participants, for each participants:

$$a \xleftarrow{\$} \mathbb{Z}_q;\ A = g^a$$

$$\begin{array}{ccc} A & & B \\ x \xleftarrow{\$} \{0,1\}^\lambda & \xrightarrow{\quad A, X = g^{H_1(x,a)} \quad} & \\ & \xleftarrow{\quad B, Y = g^{H_1(y,b)} \quad} & y \xleftarrow{\$} \{0,1\}^\lambda \end{array}$$

$$K = H(\textsc{ses}(A, B, X, Y)),\ \text{where}$$
$$\textsc{ses}(A, B, X, Y)) = (Y^a, B^{H_1(x,a)}, Y^{H_1(x,a)}, A, B)$$

Generic Protocol

Initializaton:
n participants, for each participants:

$$a \xleftarrow{\$} \mathsf{PvK};\ A = \mathsf{pv}(a)$$

$$\begin{array}{ccc}
A & & B \\
x \xleftarrow{\$} \mathsf{EphK} & \xrightarrow{\quad A, X = \mathsf{init}(x, a, r) \quad} & r' \xleftarrow{\$} U \\
r \xleftarrow{\$} U & \xleftarrow{\quad B, Y = \mathsf{resp}(y, b, r') \quad} & y \xleftarrow{\$} \mathsf{EphK}
\end{array}$$

$$K = H(\textsc{ses}(A, B, X, Y))$$

$\textsc{ses}(A, B, X, Y))$ is called the session string.

Protocol model

- A protocol is described by two roles:
  1. An initiator with actions defined by oracles
     - $\mathbf{Init}(A, B)$ starts a session (A, B, X), where A is the initiator and B is the responder, and returns X.
     - $\mathbf{Comp}(A, B, X, Y)$ completes the incomple session (A, B, X), if it exists.
  2. A responder with actions defined by oracle $\mathbf{Resp}(A, B, X)$ that completes a session (A, B, X, Y) and returns X.

Security Model

Queries that define the adversary capabilities

- Corruption: $\mathbf{Cor}(A)$ returns the private key of A
- Key reveal: $\mathbf{KeyR}(A, B, X, Y)$ reveals the session key: A checks that she has a *completed* session (A, B, X, Y). If not, she ignores the call. Otherwise, she outputs the session key.
- Ephemeral values reveal: $\mathbf{EphR}(A, B, X, -)$ reveals the ephemeral random values of a session: A checks that she has an open session $(A, B, X, -)$. If not, she ignores the call. Otherwise, she outputs the ephemeral key sampled within the above session.

Security Model

A query for defining the security game

$\mathbf{Test}(A, B, X, Y)$ depending on a randomly chosen bit b, A returns either the actual session key of the session (A, B, X, Y), or a session key drawn randomly from the session key distribution.

Unexposed sessions

Fresh sessions

A completed session (A, B, X, Y) is fresh (clean, unexposed), if the following conditions hold:

- $\mathbf{Cor}(A) \implies \mathbf{EphR}(A, B, X, Y)$ has not been asked.
- $\mathbf{Cor}(B) \implies \mathbf{Hon}(B, Y)$ and $\mathbf{EphR}(B, -, Y, -)$ has not been asked.
- neither $\mathbf{KeyR}(A, B, X, Y)$ nor $\mathbf{KeyR}(B, A, Y, X)$ has been asked.

In order to make the security definition meaningful, the adversary should only run a $\mathbf{Test}$ query on *unexposed* sessions.

Covered Security Properties

- Key-compromise impersonation: the adversary learns a long-term secret key of a party and then impersonates others to this party.
- Weak Perfect Foward Secrecy: Perfect forward secrecy means that established keys remain secret, even when the long-term keys are known after the session is completed. Krawczyk provides a generic attack for 2-message implicitly authenticated DH-baed protocols.
  Weak Perfect Serecy: the same but for passive sessions.
- It covers eCK^{ω} proposed by Cas Cremers.

Objectives

Kudla and Paterson reduce the security of an AKE protocol to the security of a similar protocol without $\mathbf{KeyR}$ and $\mathbf{Test}$ and where the adversary has to guess a session string. The reduction uses Gap DH.

Our objective:

1. Develop an automated proof of an extended version of the Kudla and Paterson reduction, in EasyCrypt. The reduction is in an extended model of CK (including $\mathbf{EphR}$), does not rely on Gap DH by construction, removes $\mathbf{KeyR}$, $\mathbf{Test}$ and $\mathbf{EphR}$, and reduces to three attacks.
2. Apply the reduction to prove security of Naxos, HMQV, etc...

A sample attack

Consider the following oracles:

1. $\mathbf{SameSeS}$: given two sessions, $\mathbf{SameSeS}$ decides whether they have the same session string
2. $\mathbf{EqS}$: given a session and a bitstring, $\mathbf{EqS}$ decides whether the latter is the session string of the former.

1. The adversay is given:
   - the long-term key a of A,
   - a message X computed with a and an ephemral value x,
   - access to oracles run by a party B,
   - $\mathbf{SameSeS}$ and $\mathbf{EqS}$.

   He wins, if he can provide a message β and a bitstring bs such that bs is the session string of (A, B, X, β).

EasyCrypt

A proof is a sequence of games related by **claims**.
Automatic or at least tool-supported verification such claims.

Rationale

- Claims are based on probabilistic statements that have a direct translation to relational Hoare judgments
- Verification of Hoare judgments reduces to the validity of formulae - verification conditions. Such verification conditions are automatically computed by EasyCrypt and submitted to SMT solvers.
- Invariant generation for oracles.
- Code-based sound transformations, e.g., eager sampling, ...

```
KG() : public_key
Init(A : part, B: part) : msg optio
Respond(B : part, A : part, X : msg) : msg option
Complete(A:part, B:part, X:msg, Y:msg) : unit
```

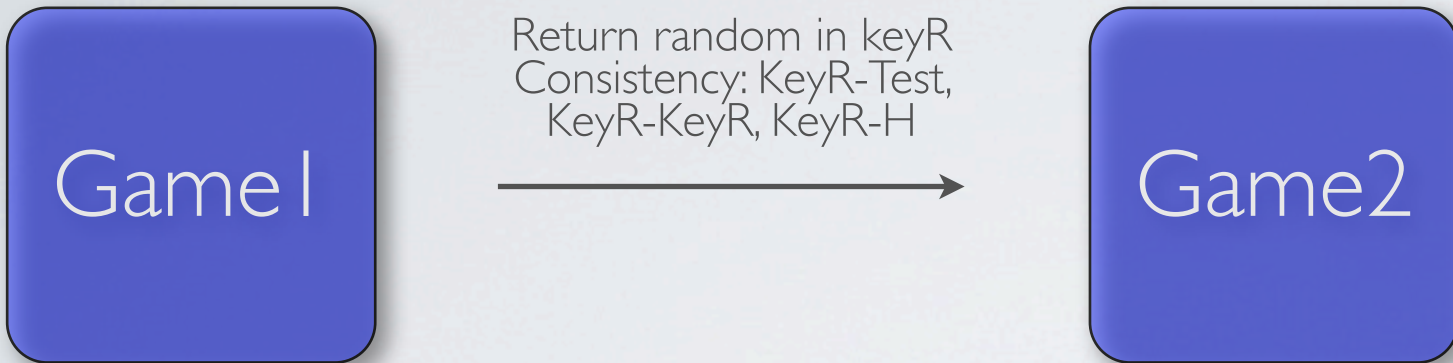
```
Hash(str : session_string) : session_key
Corrupt(A : part) : private_key option
EphKeyReveal(A : part, X : msg) :  eph_key option
KeyReveal(s : session_id) : session_key
Test(s : session_id) : session_key
```

```
fun Main () : bool = {
 var b' : bool;
 var tt : unit;
 complete_sessions = empty_map;  incomplete_sessions = empty_map;
 corrupt = empty_map;  pkey = empty_map;
 skey = empty_map;  LH  = empty_map;
 LHT  = empty_map;  seed  = empty_map;
 tested_session = empty_map;  G = empty_map;
 b = {0,1};
 b' = A();
 return (b = b');
}
```

GAME SEQUENCE

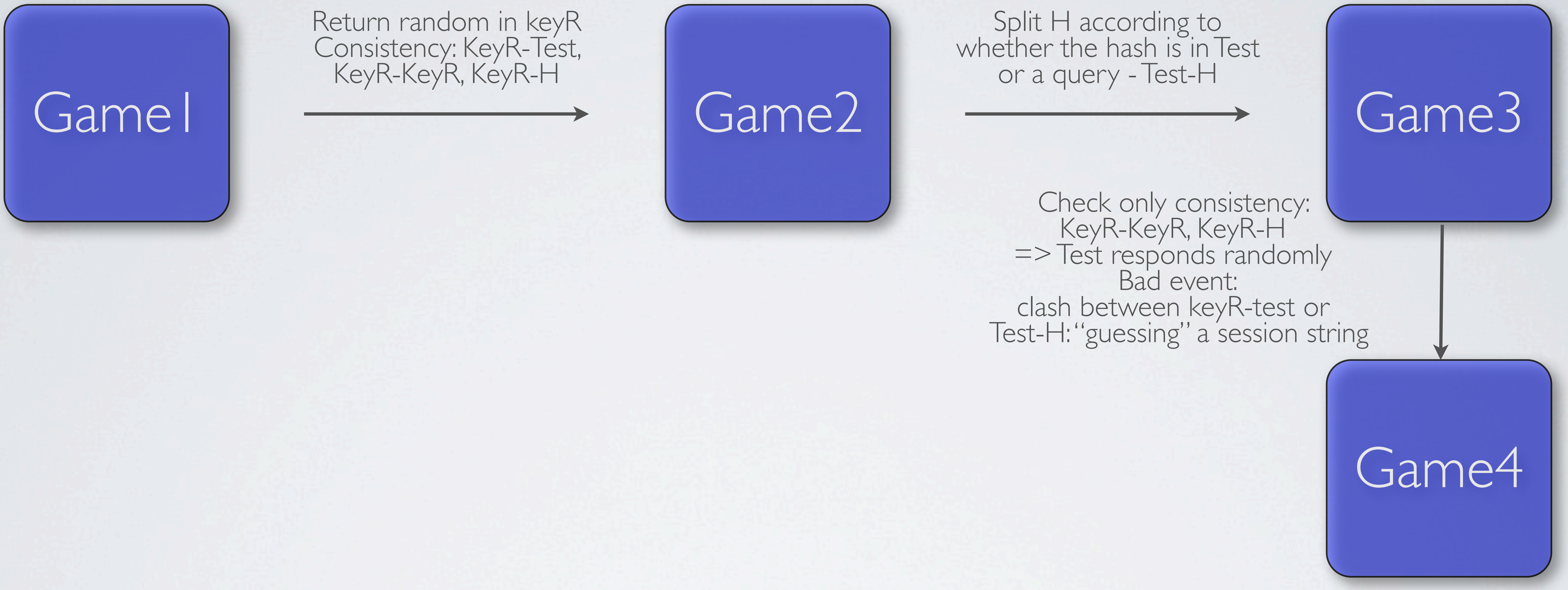
Game 1

GAME SEQUENCE



GAME SEQUENCE

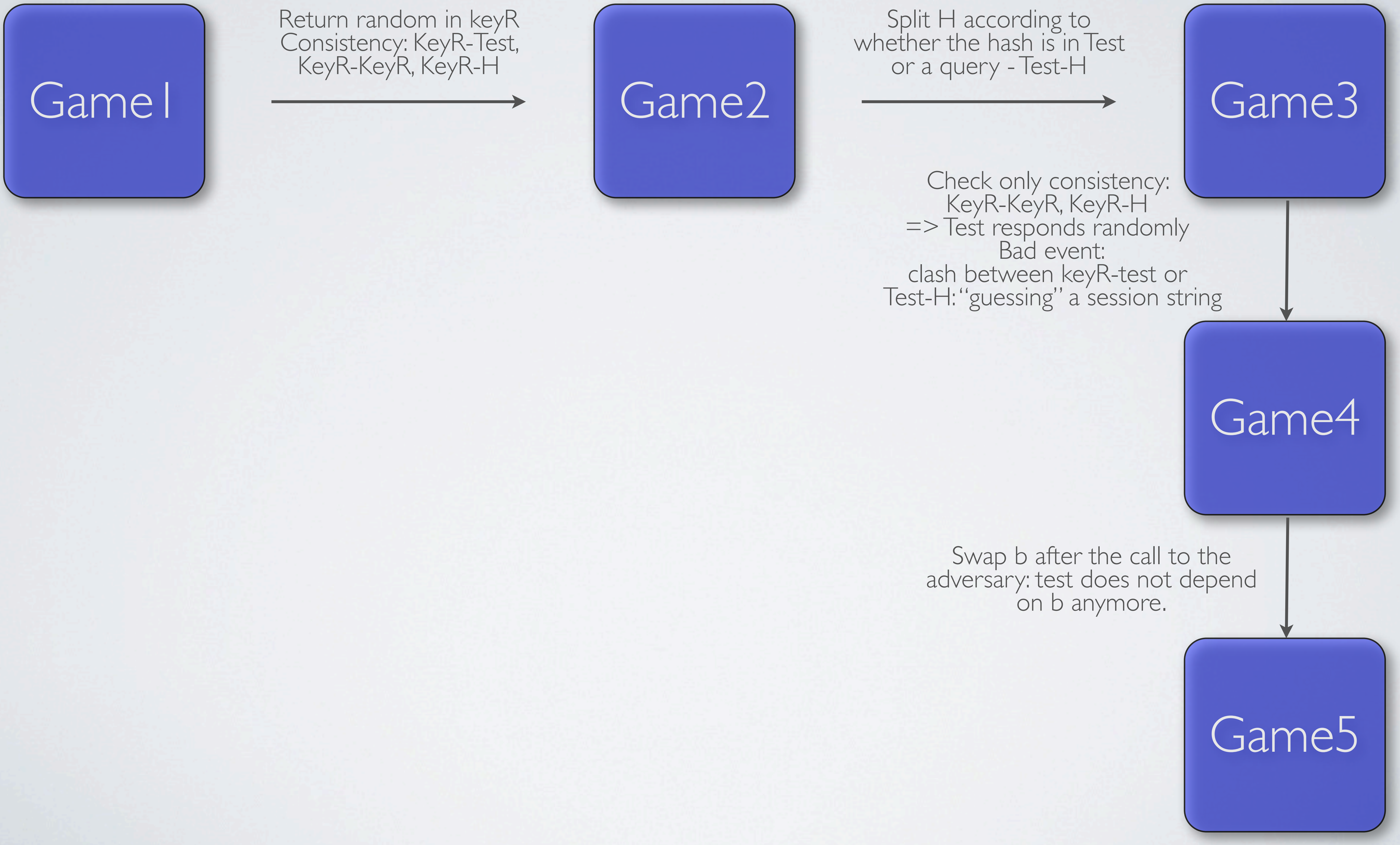
GAME SEQUENCE



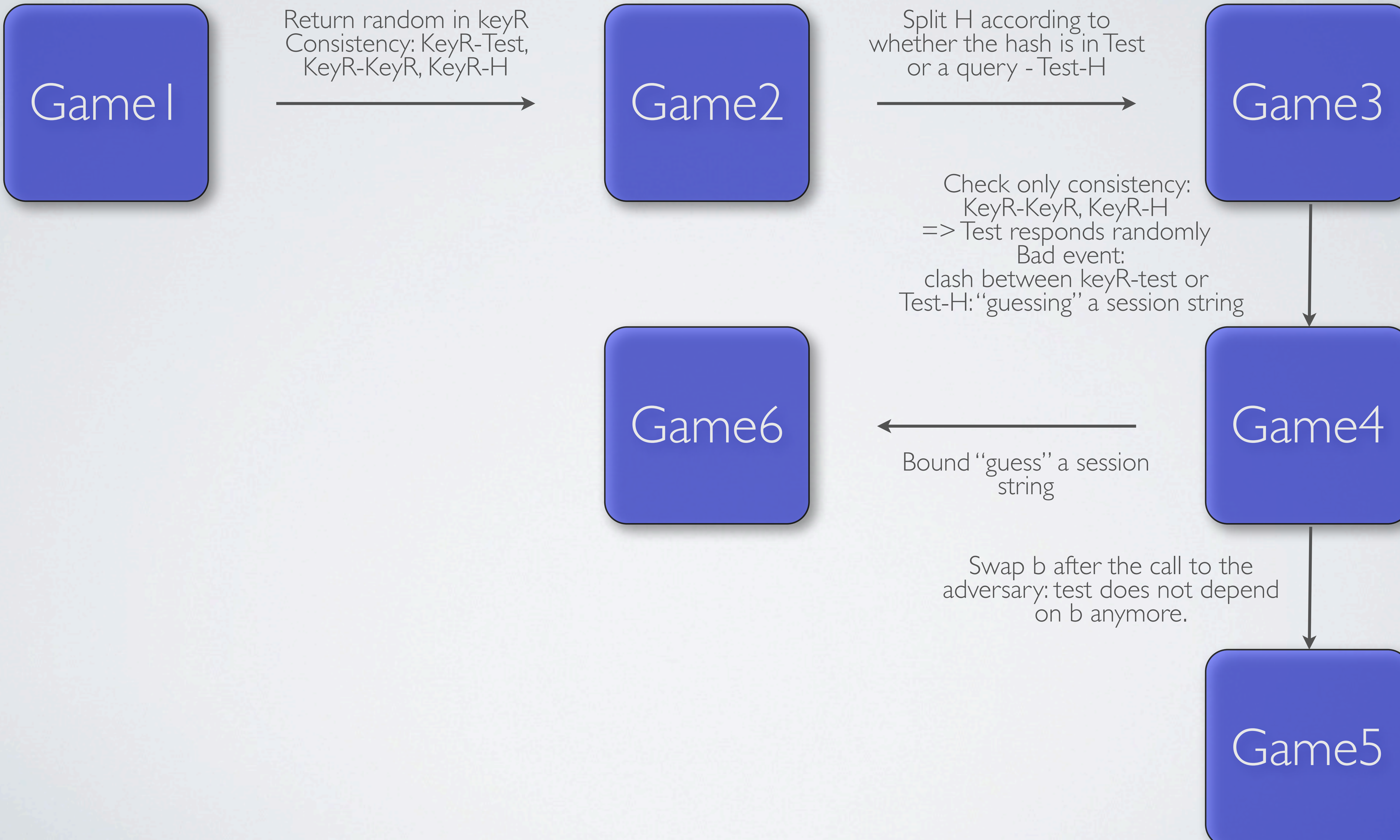
GAME SEQUENCE

GAME SEQUENCE

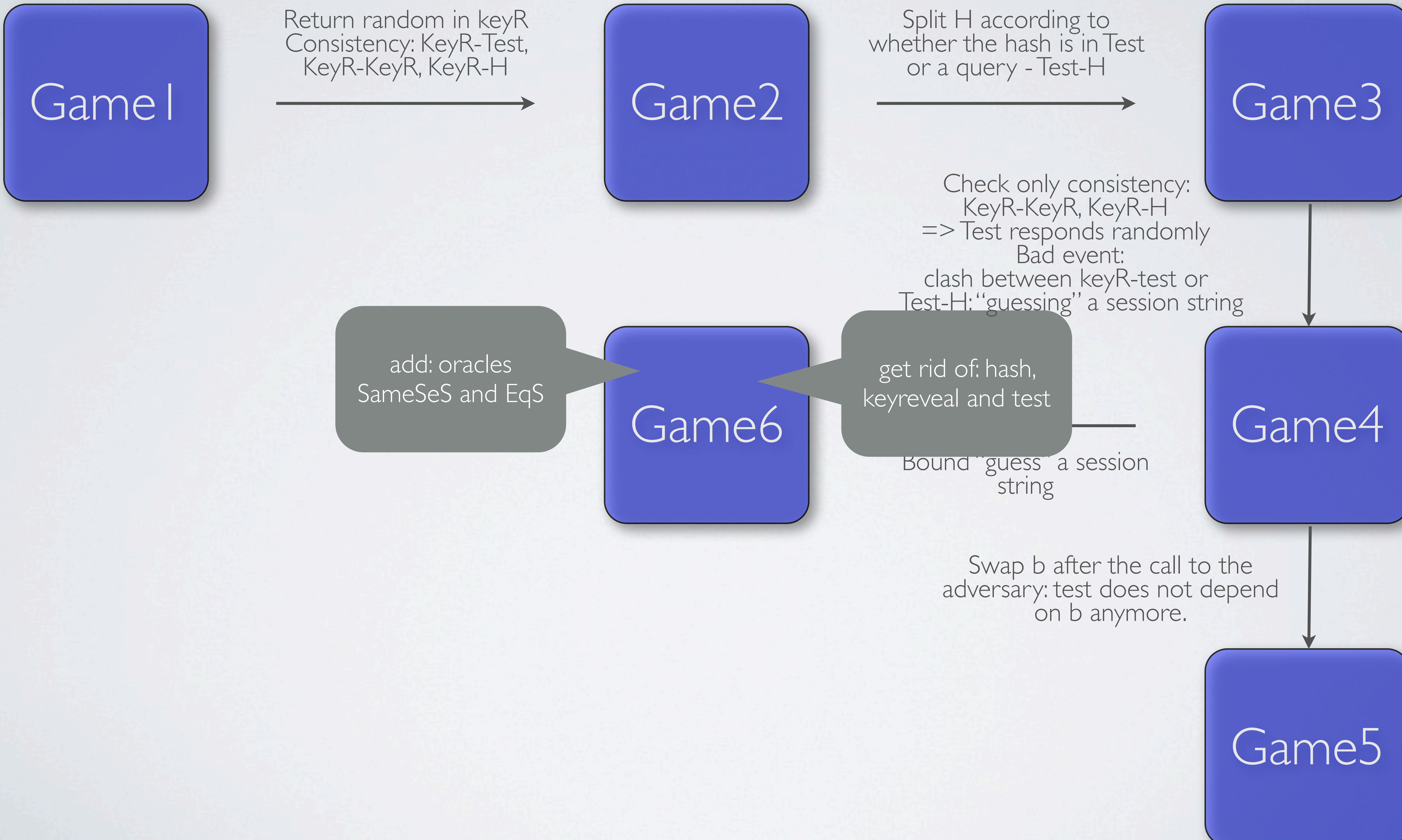
GAME SEQUENCE



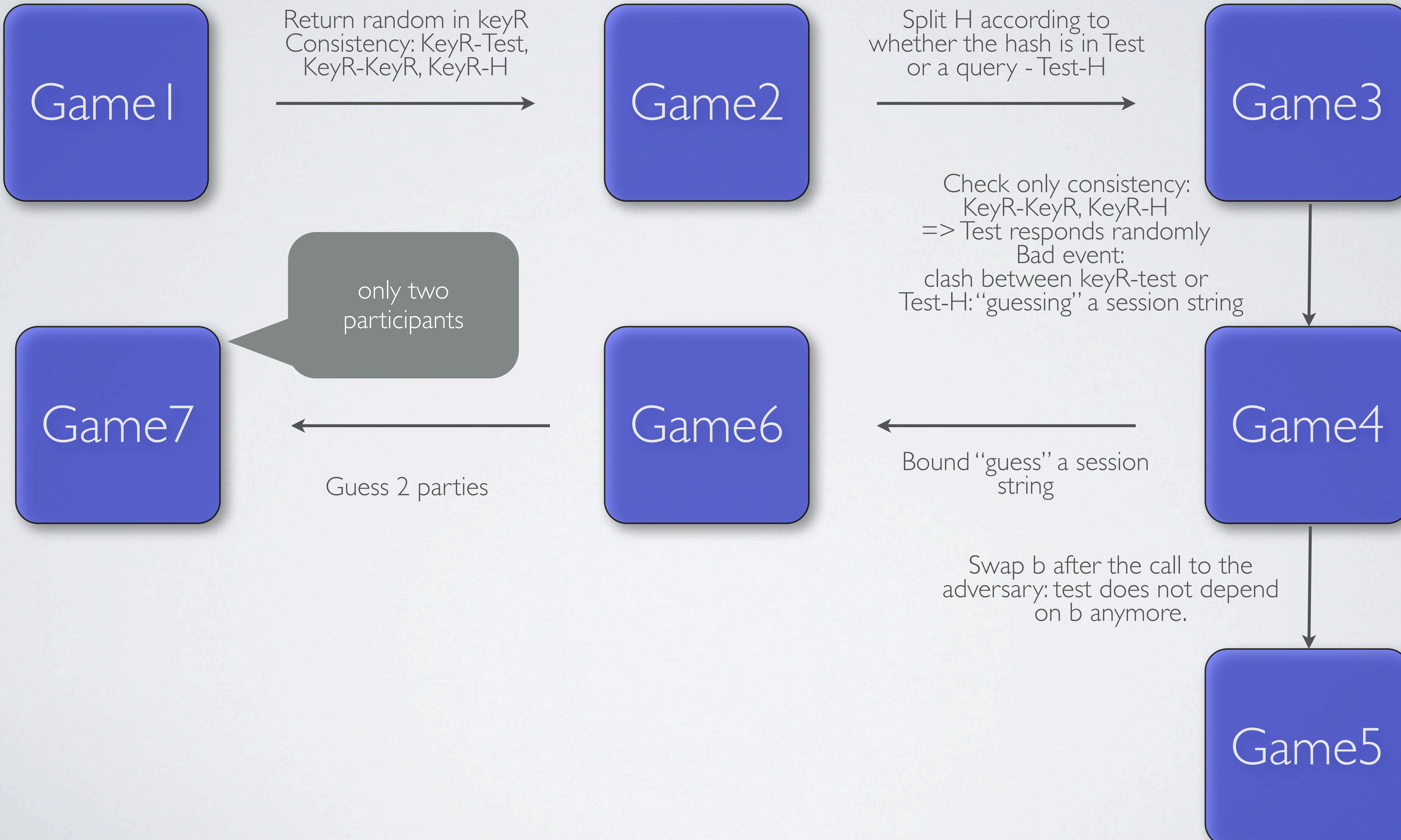
GAME SEQUENCE



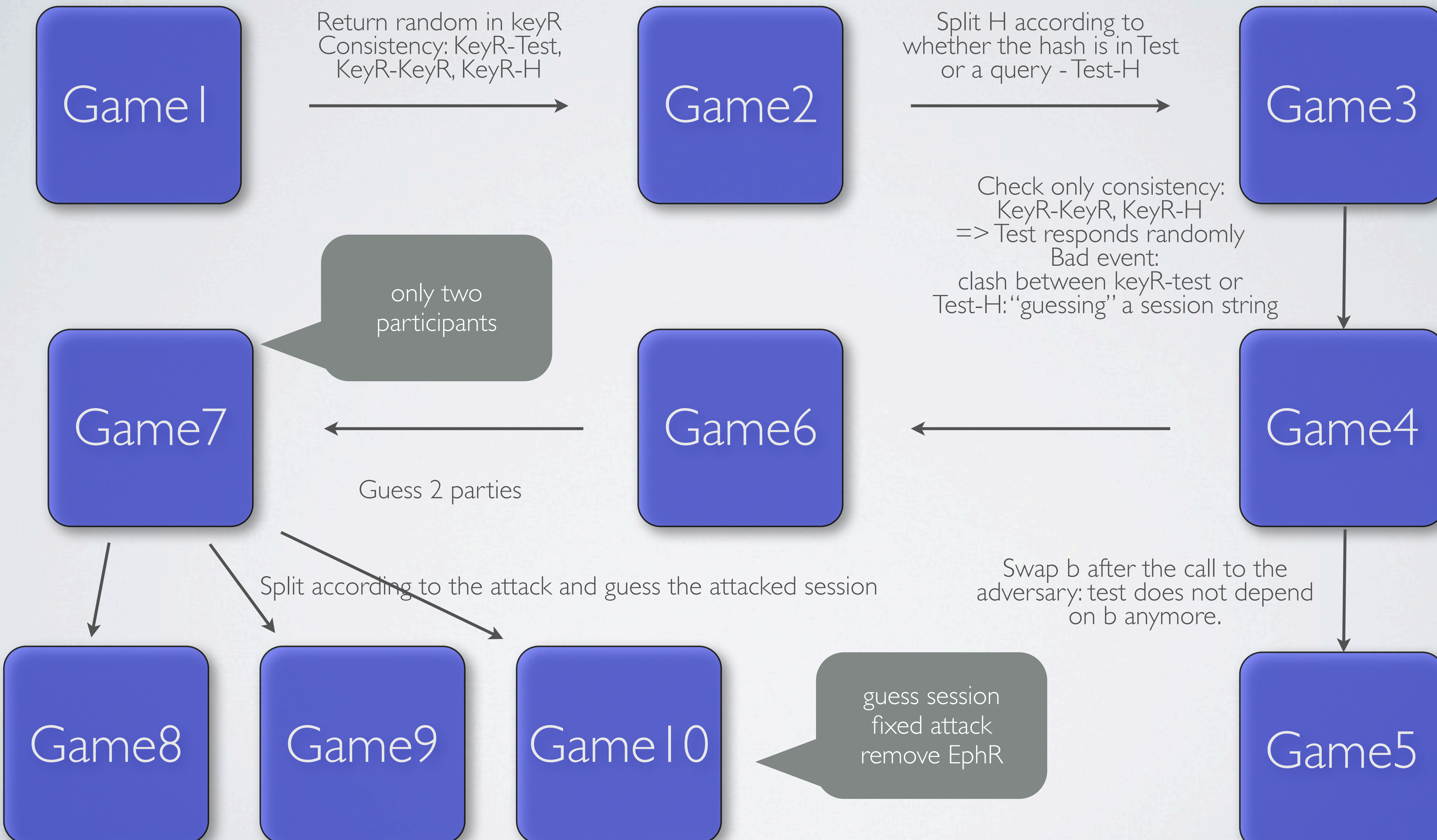
GAME SEQUENCE



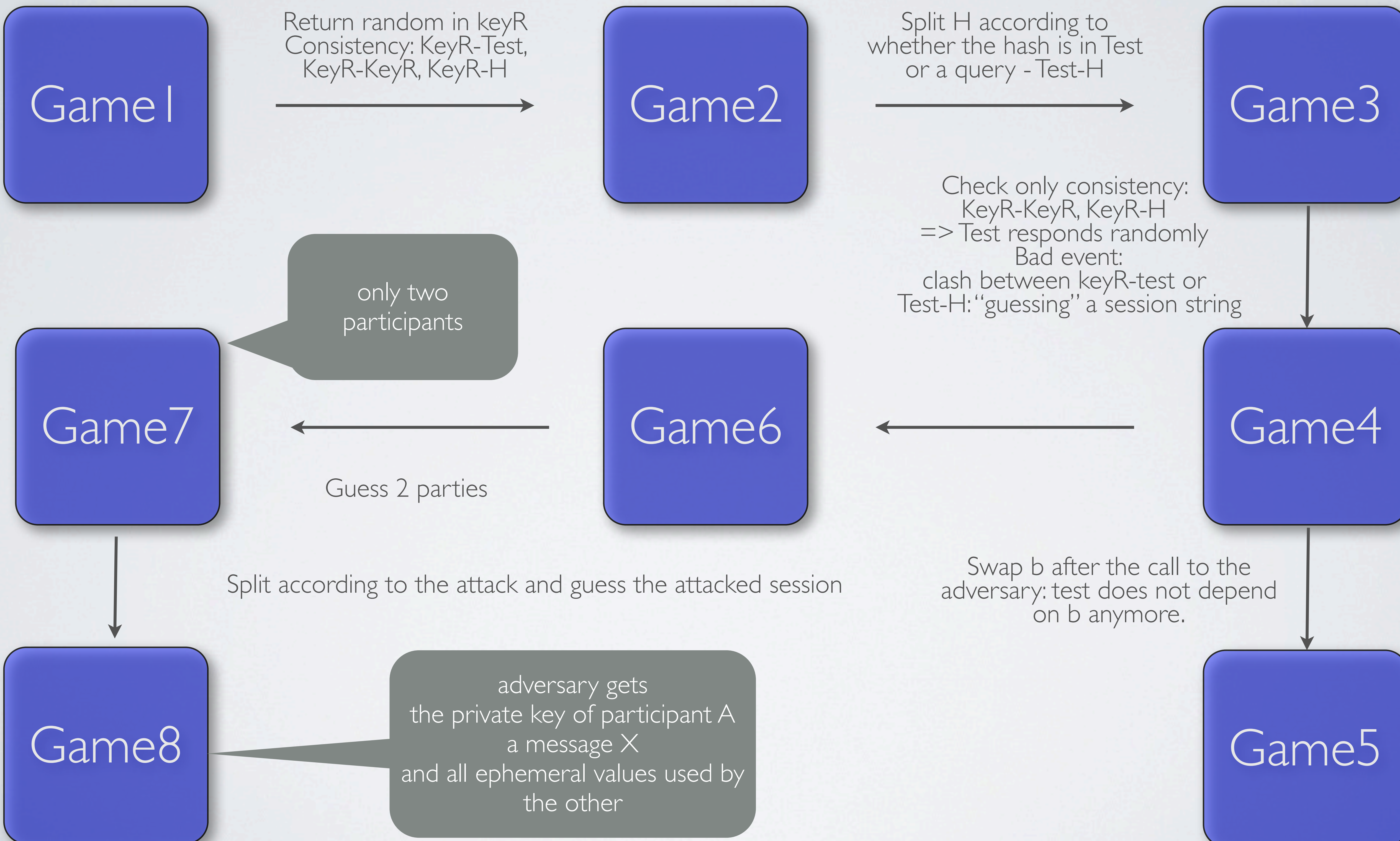
GAME SEQUENCE



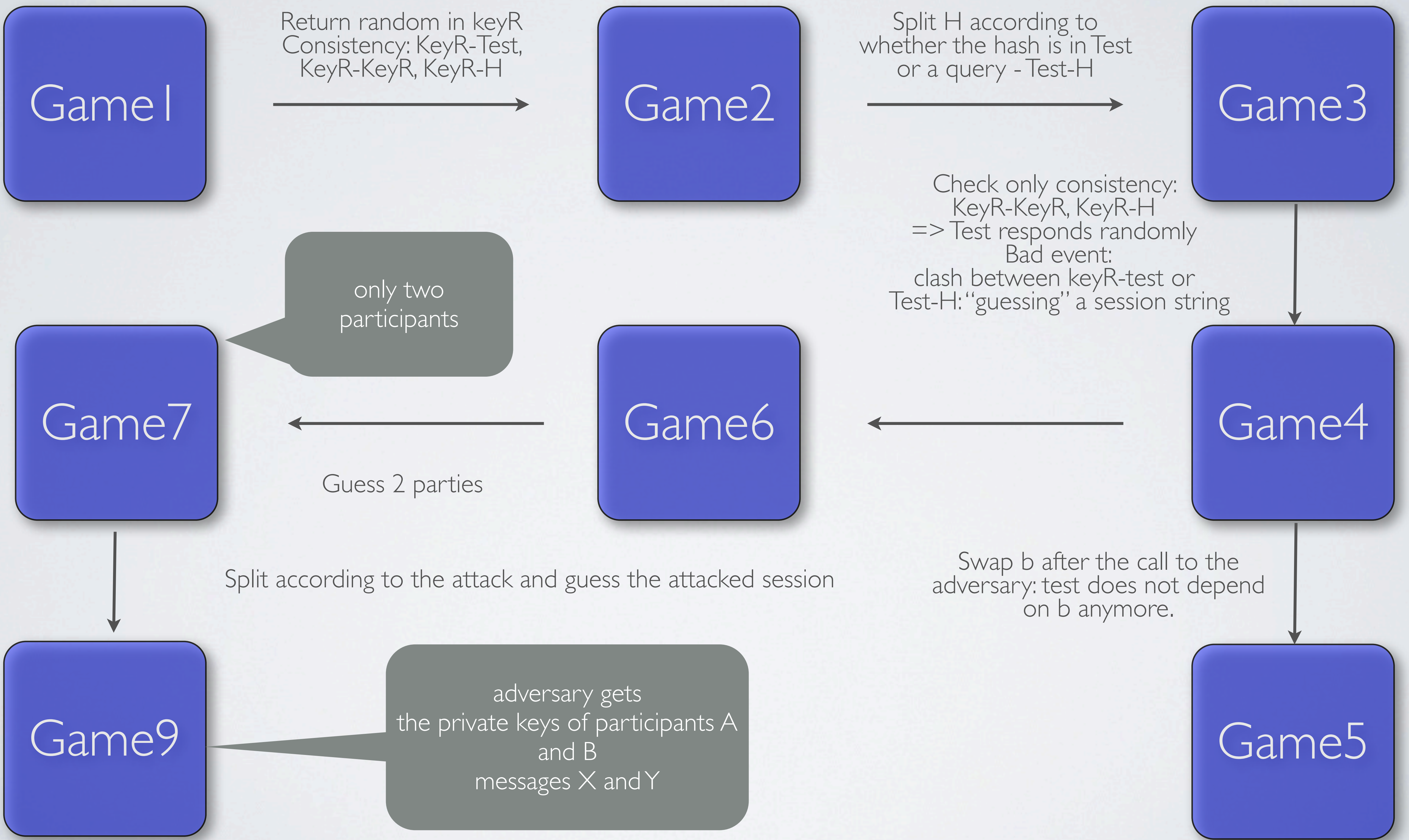
GAME SEQUENCE



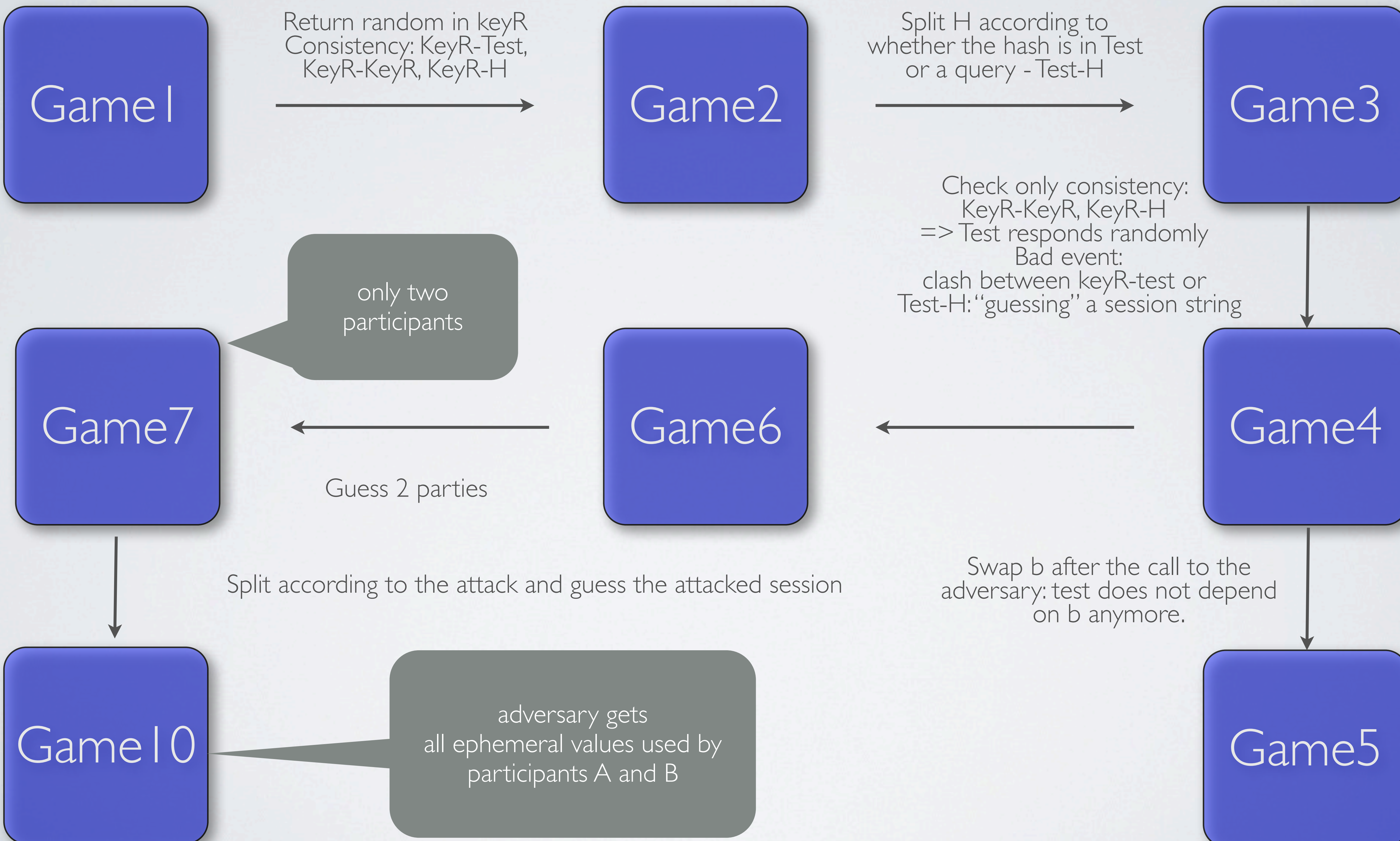
GAME SEQUENCE



GAME SEQUENCE



GAME SEQUENCE



Remarks about the model and the proof

- In order to reason about a class of protocols, our model contains abstract data types and abstract operations that are specified by axioms. Notice that abstract operations are stateless.
- Thus, the proved statement is a universal quantification over all implementations of such abstract data types.

$$
\begin{array}{l}
\forall \vec{t} \forall \vec{op},\ \mathbf{Axiom}(\vec{op}, \vec{t}) \implies \\
\quad \forall A \exists B_1 \exists B_2 \exists B_3, \\
\Pr[G_1(A, \vec{t}, \vec{op}) : b = b'] - \frac{1}{2} \leq \\
\qquad \Pr[G_8(B_1, \vec{t}, \vec{op}) : E_1] + \Pr[G_9(B_2, \vec{t}, \vec{op}) : E_2] + \\
\qquad \Pr[G_9(B_3 \vec{t}, \vec{op}) : E_3]
\end{array}
$$

- About 11 Kloc - model + invariants + proof

Initializaton:
n participants, for each participants:

$$a \xleftarrow{\$} \mathbb{Z}_q;\ A = g^a$$

$$\begin{array}{ccc} A & & B \\ x \xleftarrow{\$} \{0,1\}^\lambda & \xrightarrow{\quad A, X = g^{H_1(x,a)} \quad} & \\ & \xleftarrow{\quad B, Y = g^{H_1(y,b)} \quad} & y \xleftarrow{\$} \{0,1\}^\lambda \end{array}$$

$$K = H(\textsc{ses}(A, B, X, Y)),\ \text{where}$$
$$\textsc{ses}(A, B, X, Y)) = (Y^a, B^{H_1(x,a)}, Y^{H_1(x,a)}, A, B)$$

$\textsc{Naxos}$ uses an extra hash function H_1. Which is not part of the generic model.

Solution: reduce security of $\textsc{Naxos}$ to security of $\textsc{Naxos}$' by "internalizing" H_1.

$$
\begin{array}{ccc}
A & & B \\
x \xleftarrow{\$} \{0,1\}^\lambda & \xrightarrow{\quad A, X = g^r \quad} & r' \xleftarrow{\$} \mathbb{Z}_q \\
r \xleftarrow{\$} \mathbb{Z}_q & \xleftarrow{\quad B, Y = g^{r'} \quad} & y \xleftarrow{\$} \{0,1\}^\lambda
\end{array}
$$

$$K = H(\textsc{ses}(A, B, X, Y)), \text{ where}$$

$$\textsc{ses}(A, B, X, Y)) = (Y^a, B^r, Y^r, A, B)$$

For all adversaries A there is an adversary B s.t.

$$
\begin{aligned}
&\Pr[G_1'(A, \textsc{Naxos}) : b' = b] = \\
&\quad \Pr[G_1(B, \textsc{Naxos}')b' = b] + \Pr[G_1(B, \textsc{Naxos}) : \text{ Guess fresh } a \text{ or } x]
\end{aligned}
$$

Using the same generic proof, we can bound the probability of

$$\Pr[G_1(B, \textsc{Naxos}) : \text{ Guess fresh } a \text{ or } x]$$

with respect to Gap Discrete Log.
Gap Discrete Log:
Given g^a, compute a while having access to a restricted DDH oracle:

$$(g^x, z) : \; z \stackrel{?}{=} g^{ax}$$

Conclusion

- An automated proof in EasyCrypt of a modular reduction of the key security of hashed key exchange protocols to simplified session string unforgery.
- Applications: HMQV, NAXOS, a new famity of protocols based on twin DH we designed.

Several challenges for EasyCrypt

- Introduction of an instantiation mechanism.
- Automatic invariant generation.
- Better handling of verification conditions.
- Improvement of the undelying logic - exploiting CIL (Computational Indistinguishability Logic)