Backdoored Pseudorandom Generators

Yevgeniy Dodis, **Chaya Ganesh**, Alexander Golovnev, Ari Juels, Thomas Ristenpart

Subversion of Cryptography

- Black-box implementations are target of subversion into subliminal channels - “Kleptography” (Young and Yung)

Subversion of Cryptography

- Black-box implementations are target of subversion into subliminal channels - “Kleptography” (Young and Yung)
- Recent Snowden revelations - Surveillance, Backdoors in cryptographic standards and software.

Subversion of Cryptography

- Black-box implementations are target of subversion into subliminal channels - “Kleptography” (Young and Yung)
- Recent Snowden revelations - Surveillance, Backdoors in cryptographic standards and software.
- Backdoored NIST standard - Dual EC PRG

Subversion of Cryptography

- Black-box implementations are target of subversion into subliminal channels - “Kleptography” (Young and Yung)
- Recent Snowden revelations - Surveillance, Backdoors in cryptographic standards and software.
- Backdoored NIST standard - Dual EC PRG
- Subversion of TLS encryption (Checkoway et al)

Subversion of Cryptography

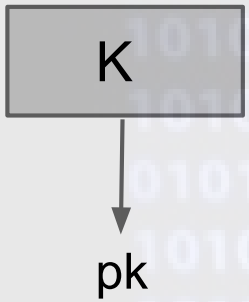
- Black-box implementations are target of subversion into subliminal channels - “Kleptography” (Young and Yung)
- Recent Snowden revelations - Surveillance, Backdoors in cryptographic standards and software.
- Backdoored NIST standard - Dual EC PRG
- Subversion of TLS encryption (Checkoway et al)
- This work - Backdoored Pseudorandom Generators

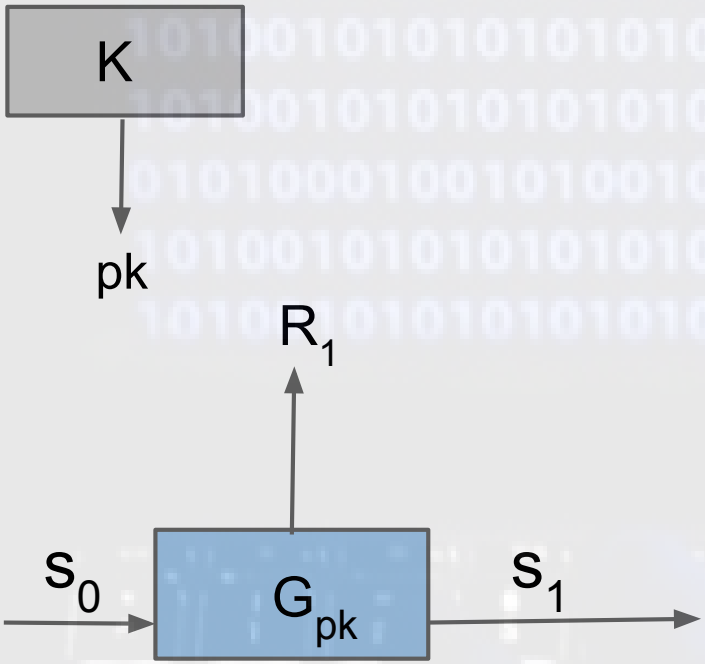
Pseudorandom Generator

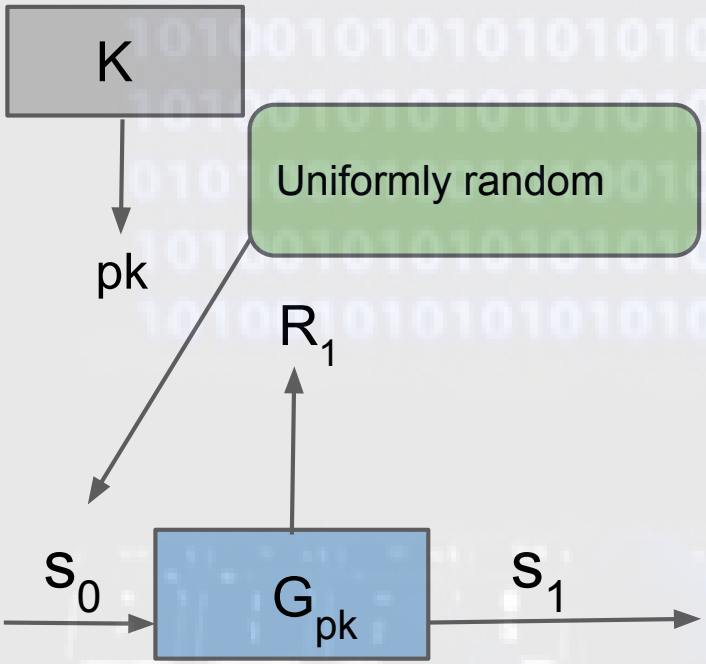
- Stretches a short uniform random string into a long sequence of pseudorandom bits
- **Security**: A PRG is secure when no adversary can distinguish between its outputs and random bits.

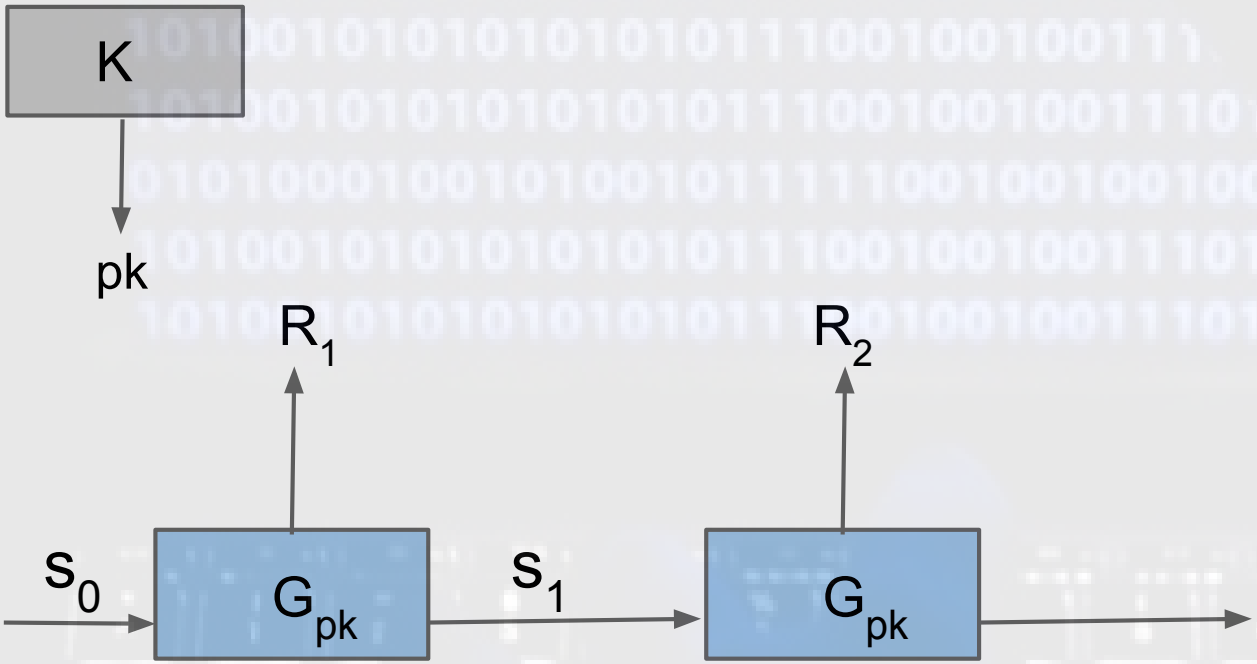
PRG Family

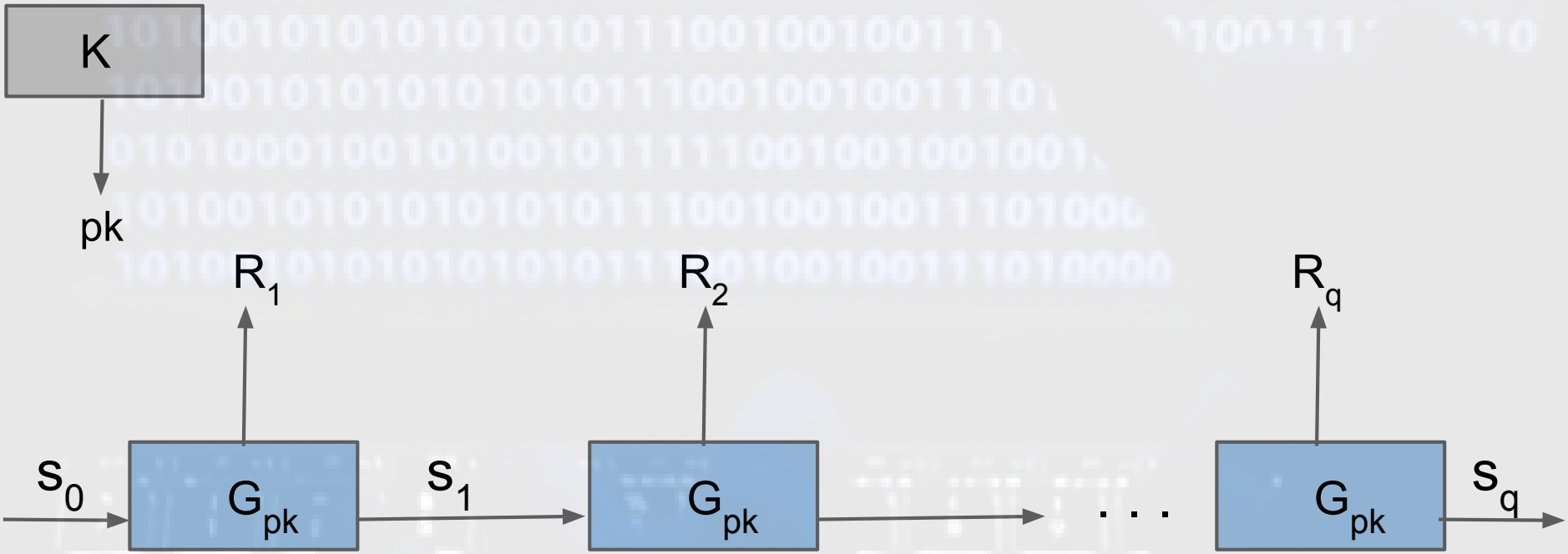
- We consider a family of PRGs - for efficiency
- A public parameter (like IV) designates a family
- The public parameters pk picked once and are "innocent" looking, typically random $pk \equiv \mathcal{U}$
- Each algorithm $\mathrm{G}_{pk} : \mathcal{S} \to \{0,1\}^n \times \mathcal{S}$ maps an input called the state to an n-bit output and a new state


K
pk


K
pk
R_1
s_0
G_{pk}
s_1

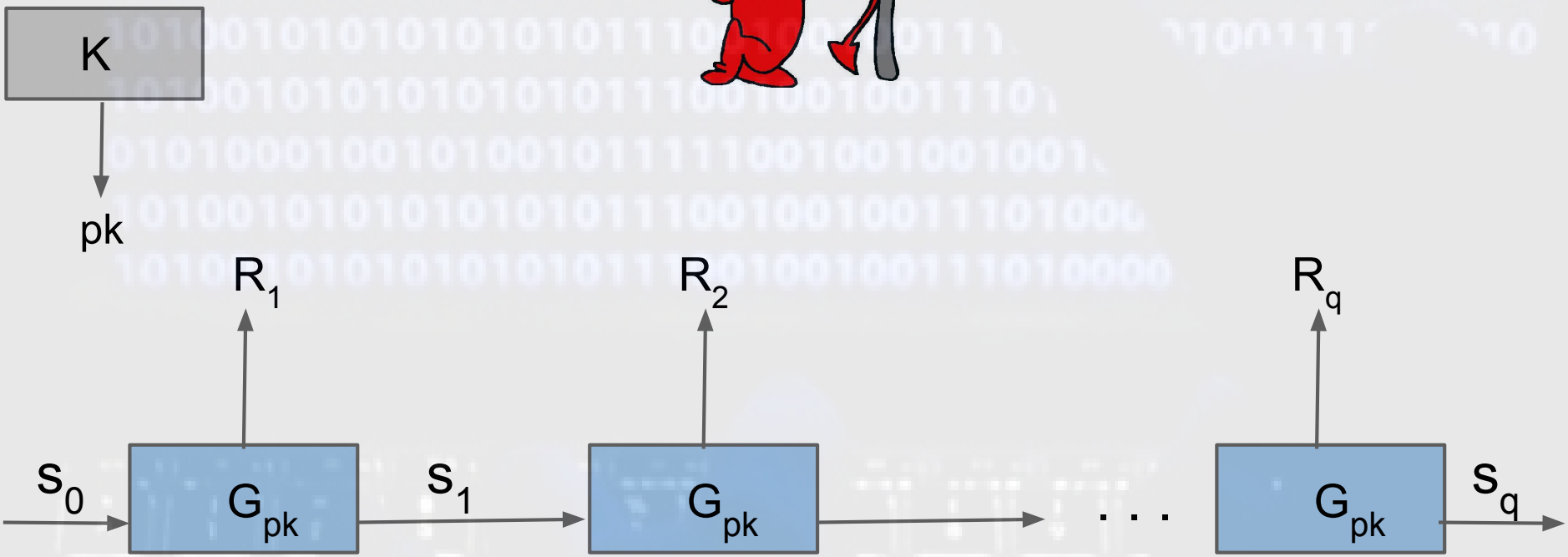

K
pk
Uniformly random
R_1
s_0
G_{pk}
s_1


K
pk
R_1
R_2
s_0
G_{pk}
s_1
G_{pk}


K
pk
R_1
R_2
R_q
s_0
G_{pk}
s_1
G_{pk}
. . .
G_{pk}
s_q

Adversary

A



Adversary

A

K

pk

$\mathcal{U}_1$

$\mathcal{U}_2$

$\mathcal{U}_q$

Adversary
A

K

pk

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group
- The public key is a random $pk = y$ from the group G

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group
- The public key is a random $pk = y$ from the group G
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group
- The public key is a random $pk = y$ from the group G
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- Can be proven secure under the DDH assumption

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group
- The public key is a random $pk = y$ from the group G
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- Can be proven secure under the DDH assumption
  - Detail - Encode group elements as bit strings

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group
- The public key is a random $pk = y$ from the group G
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- Can be proven secure under the DDH assumption
  - Detail - Encode group elements as bit strings
  - Not important for the purposes of this talk

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group
- The public key is a random $pk = y$ from the group G
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- Can be proven secure under the DDH assumption
  - Detail - Encode group elements as bit strings
  - Not important for the purposes of this talk
- All good?

Simplified Dual EC PRG

- Let G be a group and g be a generator of the group
- The public key is a random $pk = y$ from the group G
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- Can be proven secure under the DDH assumption
  - Detail - Encode group elements as bit strings
  - Not important for the purposes of this talk
- All good?
- No! (Shumow and Ferguson 2007)

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

for x chosen at random from the group

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

for x chosen at random from the group

- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

  for x chosen at random from the group
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- **Attack**

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

  for x chosen at random from the group
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- **Attack**
  - Adversary A in possession of x sees one output r_{i+1}

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

  for x chosen at random from the group
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- **Attack**
  - Adversary A in possession of x sees one output r_{i+1}
  - Can recover s_{i+1} by computing $s_{i+1} = (r_{i+1})^x = y^{s_i}$

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

  for x chosen at random from the group
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- **Attack**
  - Adversary A in possession of x sees one output r_{i+1}
  - Can recover s_{i+1} by computing $s_{i+1} = (r_{i+1})^x = y^{s_i}$
  - All subsequent outputs predictable from current state

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

  for x chosen at random from the group
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- **Attack**
  - Adversary A in possession of x sees one output r_{i+1}
  - Can recover s_{i+1} by computing $s_{i+1} = (r_{i+1})^x = y^{s_i}$
  - All subsequent outputs predictable from current state
- Dual EC PRG works as above (drops the last 16 bits)

Simplified Dual EC PRG

- Suppose the public key y is chosen as follows:

$$y = g^x$$

  for x chosen at random from the group
- The PRG works as follows:

$$\mathrm{G}_{pk}(s_i) = (r_{i+1}, s_{i+1}) = (g^{s_i}, y^{s_i})$$

- **Attack**
  - Adversary A in possession of x sees one output r_{i+1}
  - Can recover s_{i+1} by computing $s_{i+1} = (r_{i+1})^x = y^{s_i}$
  - All subsequent outputs predictable from current state
- Dual EC PRG works as above (drops the last 16 bits)
- Motivates the formal study of Backdoored Pseudorandom generators

Summary of results

- Definitional framework of Backdoored PRGs.
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts.
- Investigate countermeasures to BPRGs - immunizers.
  - (In)effectiveness of countermeasures
  - Provably secure solution

Summary of results

- Definitional framework of Backdoored PRGs
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts.
- Investigate countermeasures to BPRGs - immunizers.
  - (In)effectiveness of countermeasures
  - Provably secure solution

Backdoored PRG

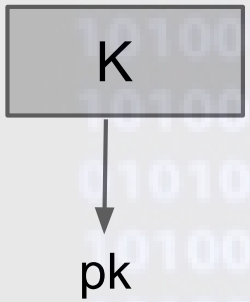
Intuition:

- Behaves like a good PRG to an honest user, but...
- Knowledge of trapdoor information compromises the security of the PRG.

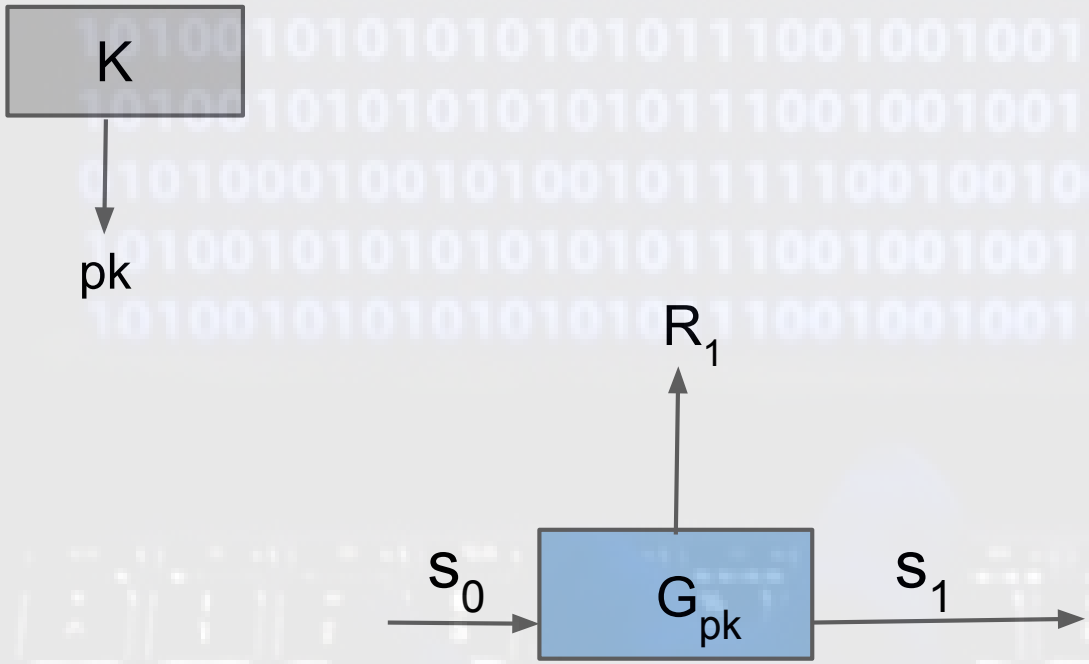
Formally,

- A triple of algorithms $(\mathrm{K}, \mathrm{G}, \mathrm{A})$, where $\mathrm{K}(\$) \to (pk, sk)$
- **Standard PRG security** Ignoring *sk*, the pair (K, G) is a PRG
- **Subversion** The third algorithm A (the adversary) co-designed with the rest of the scheme, uses the trapdoor *sk* output by K to violate security of the PRG.

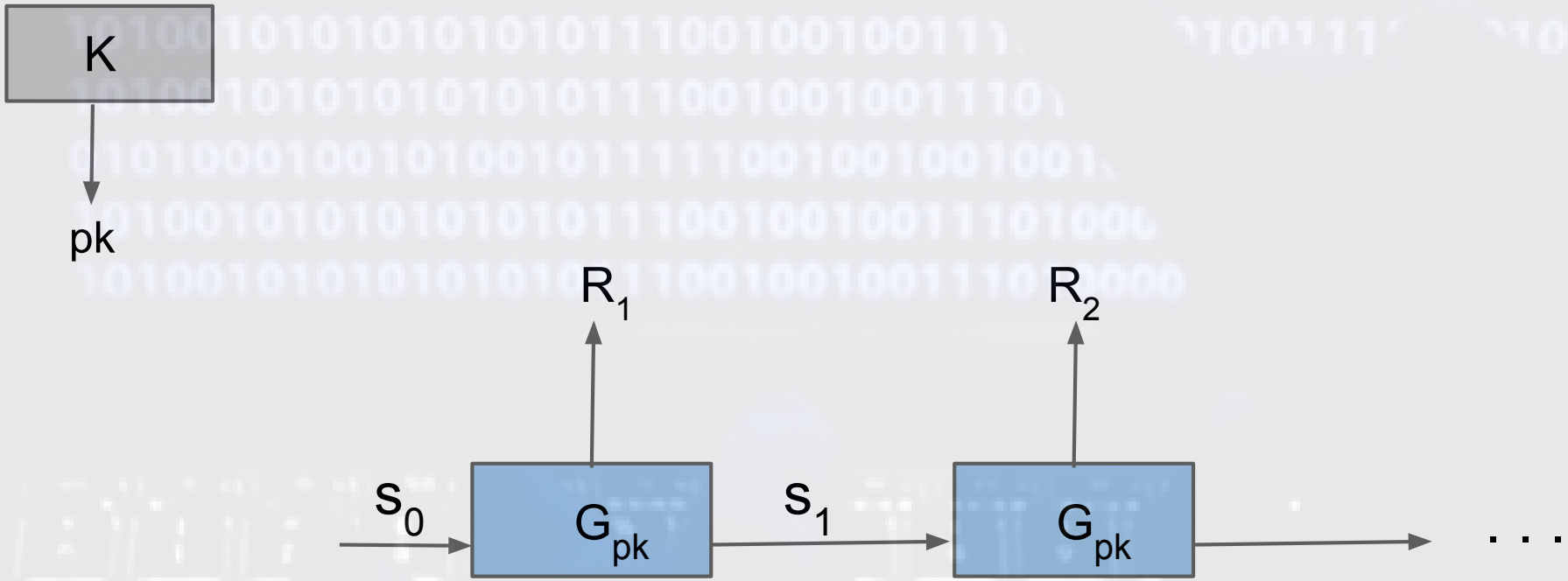
Normal Operation



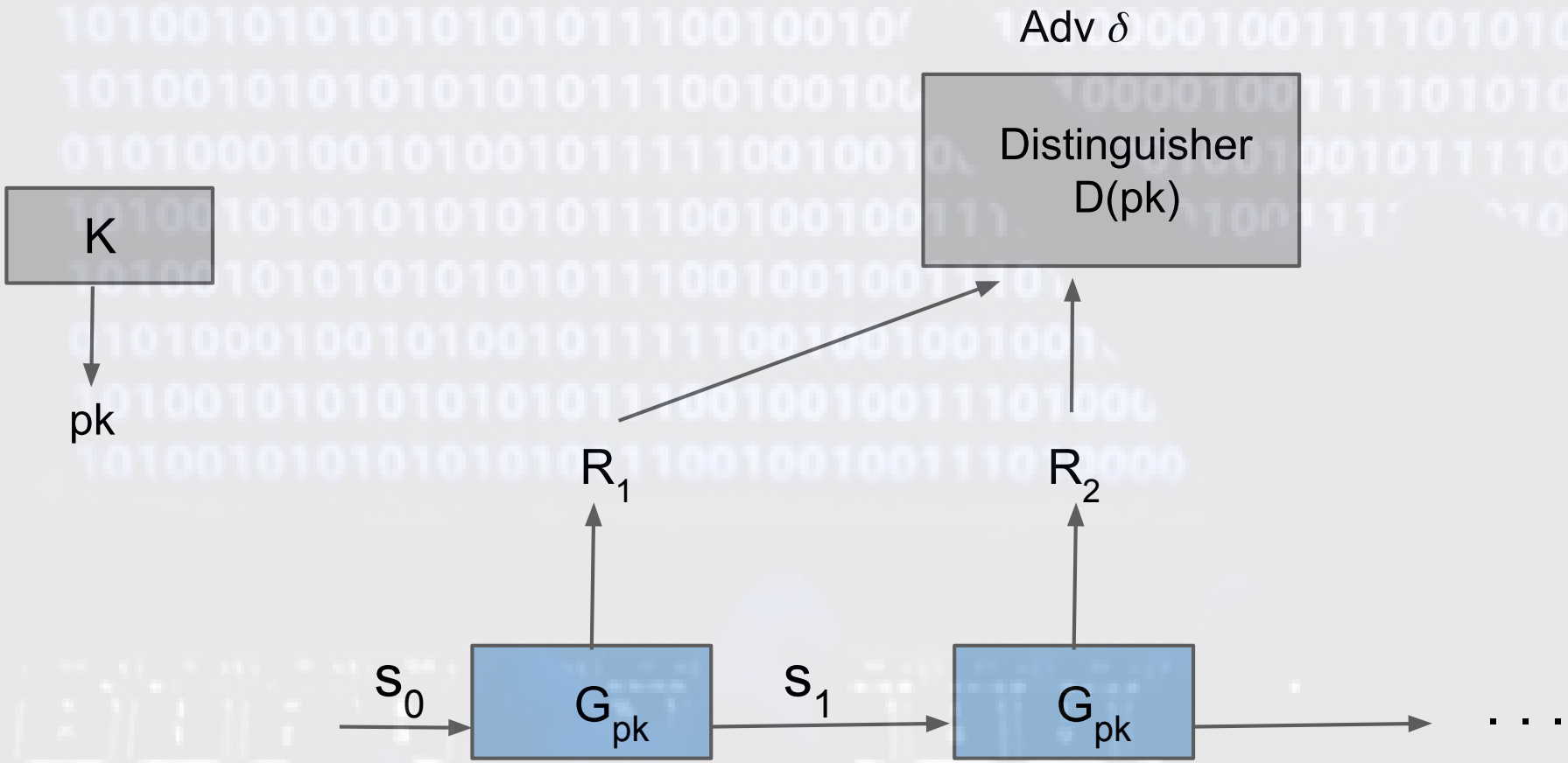
Normal Operation



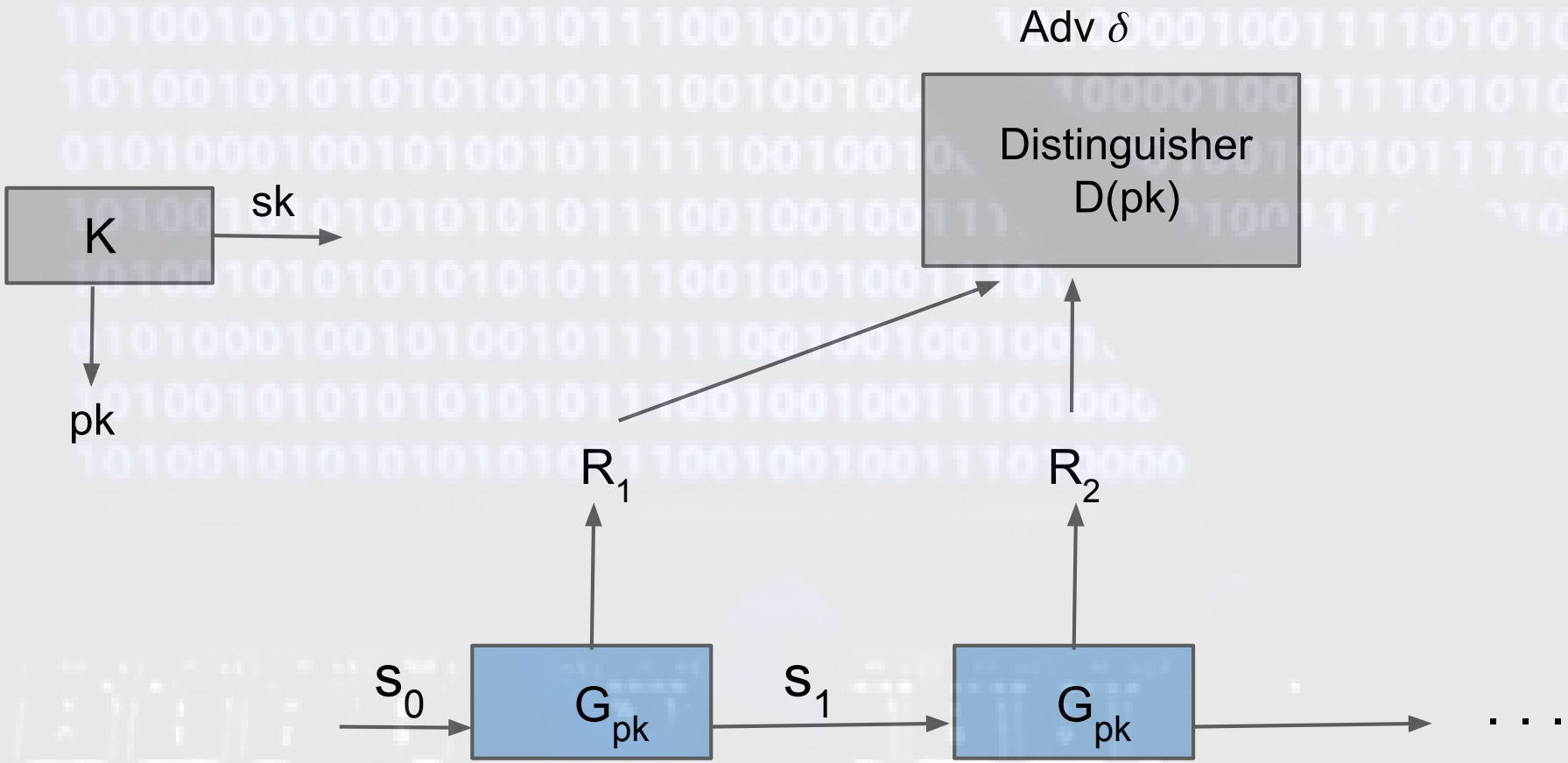
Normal Operation



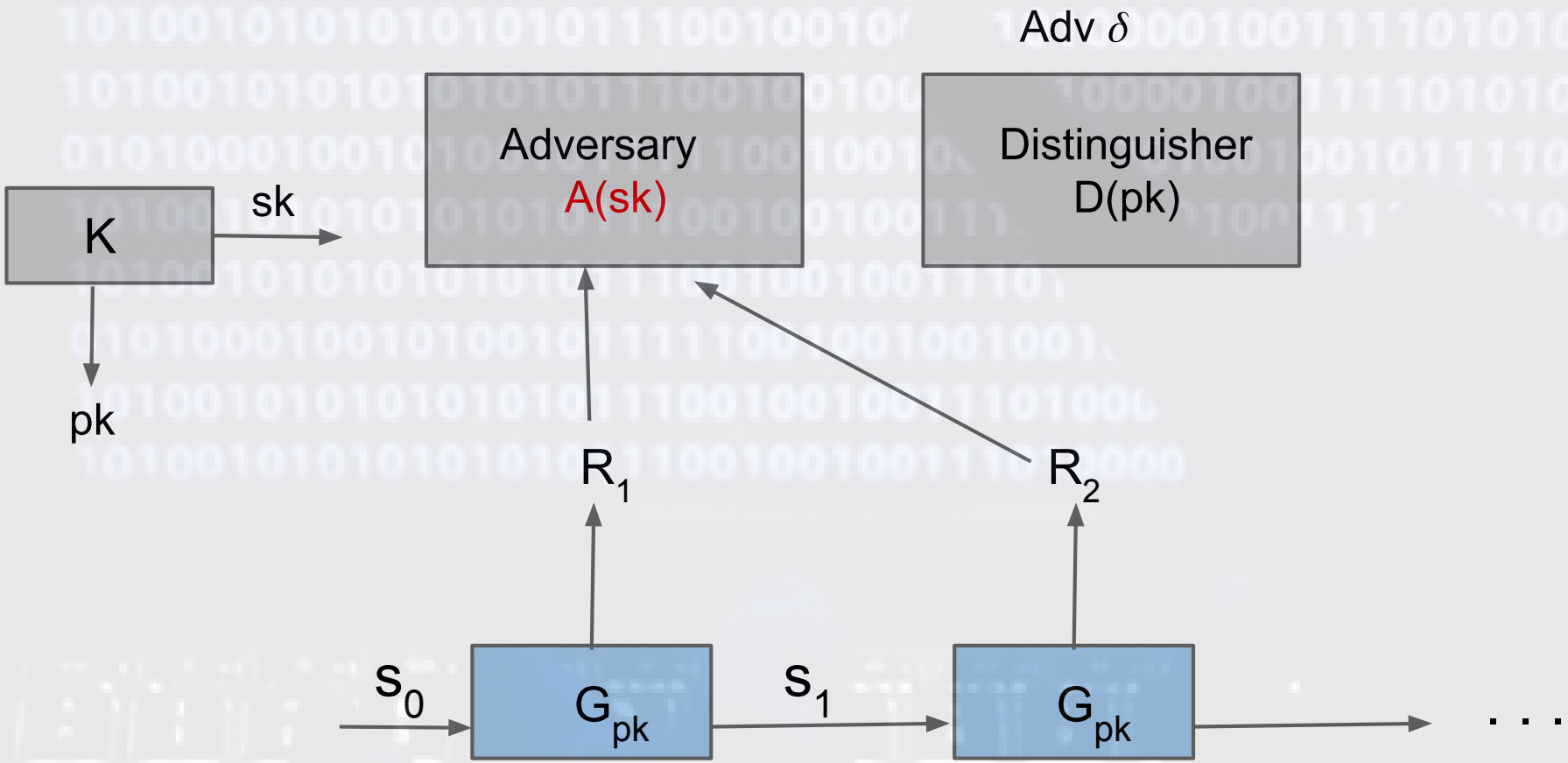
Normal Operation



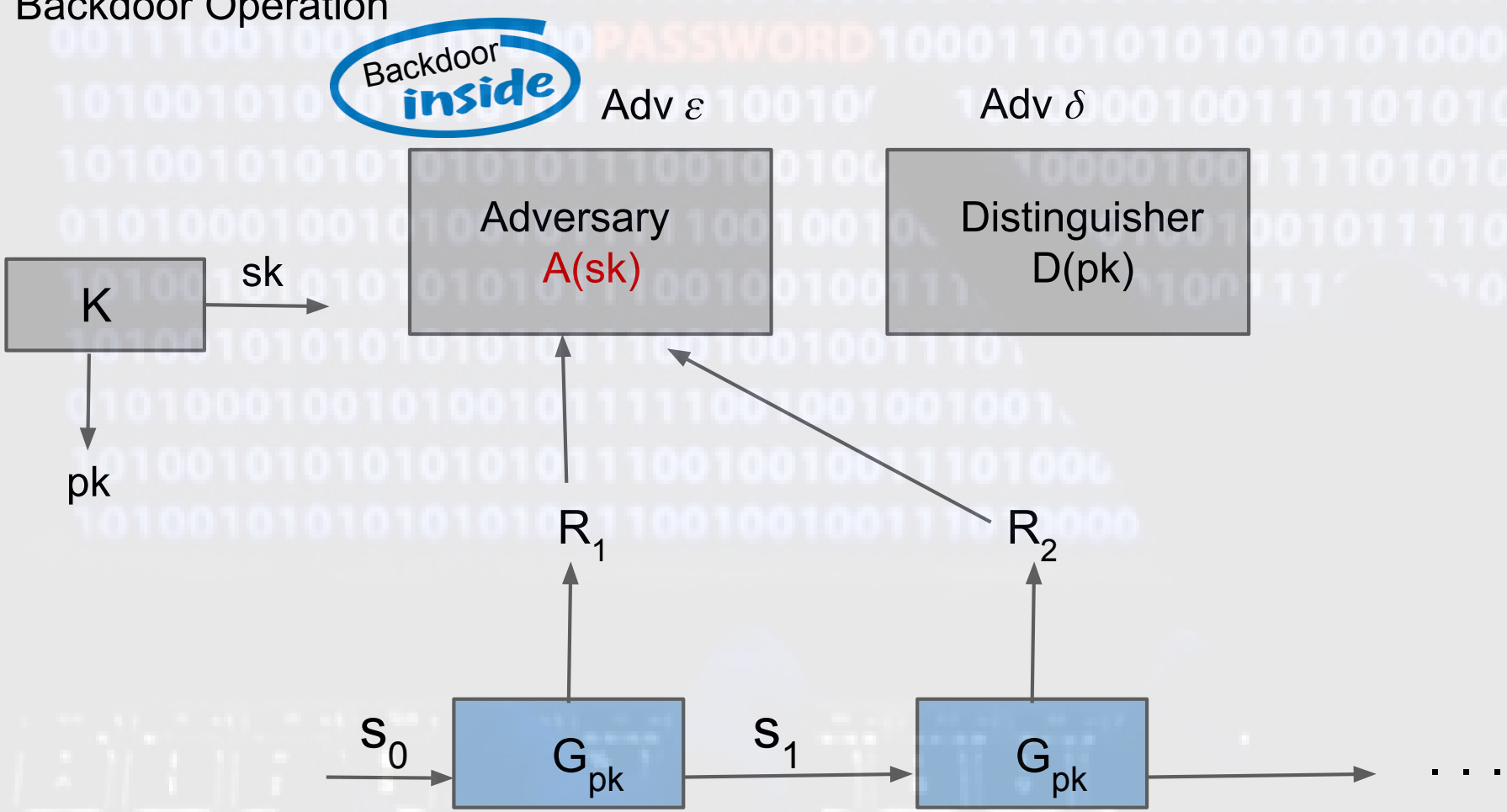
Backdoor Operation



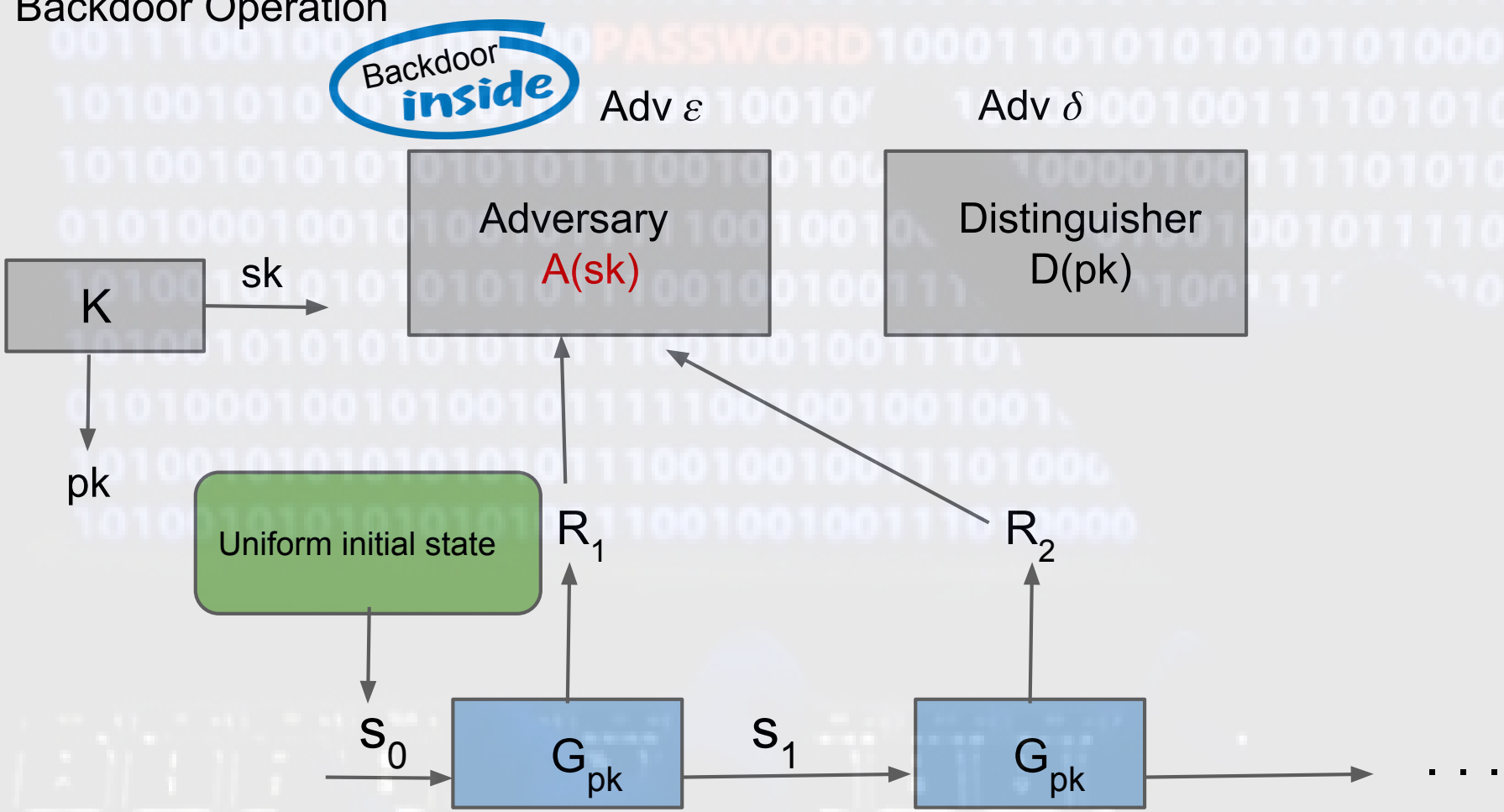
Backdoor Operation



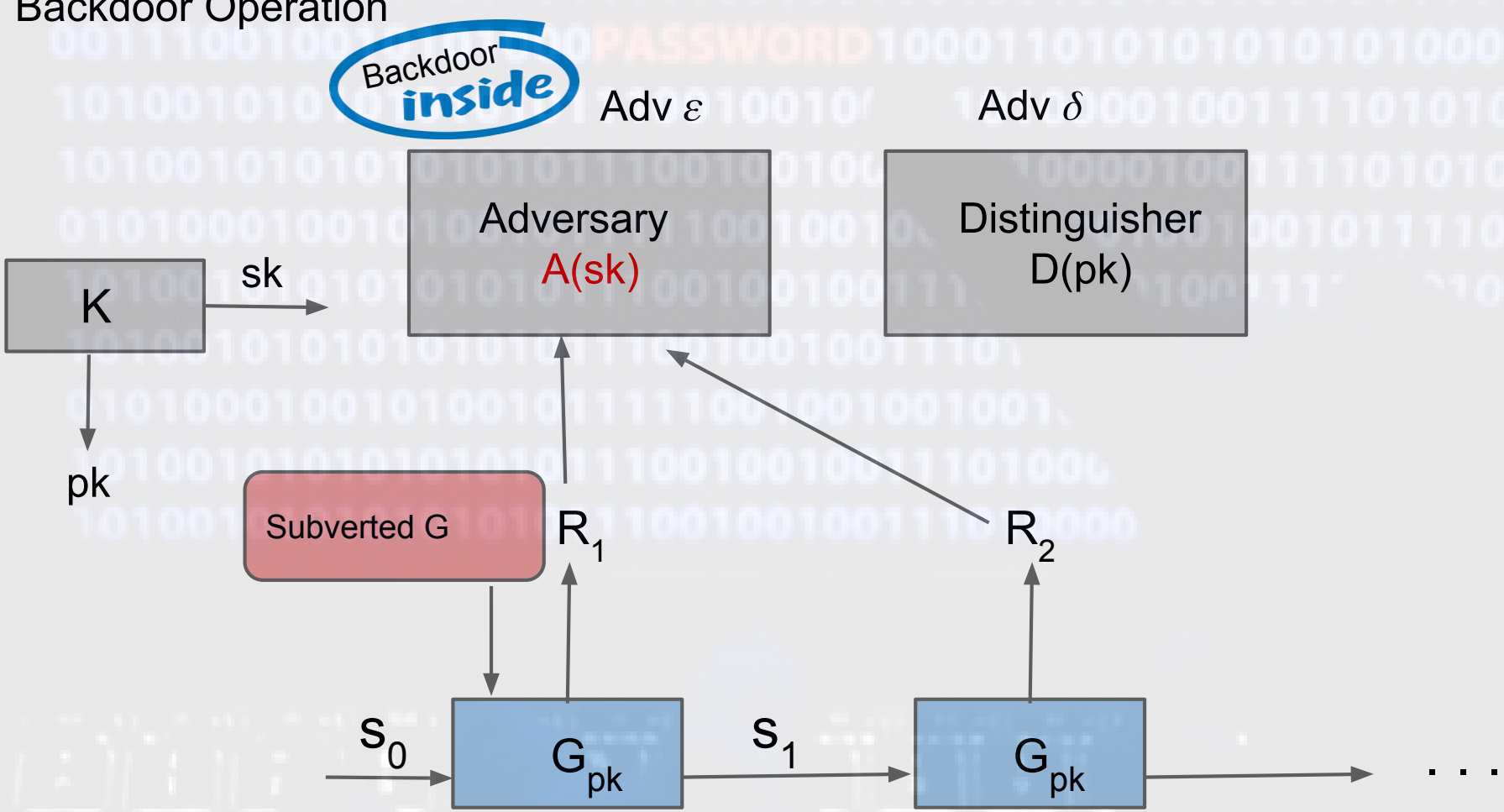
Backdoor Operation



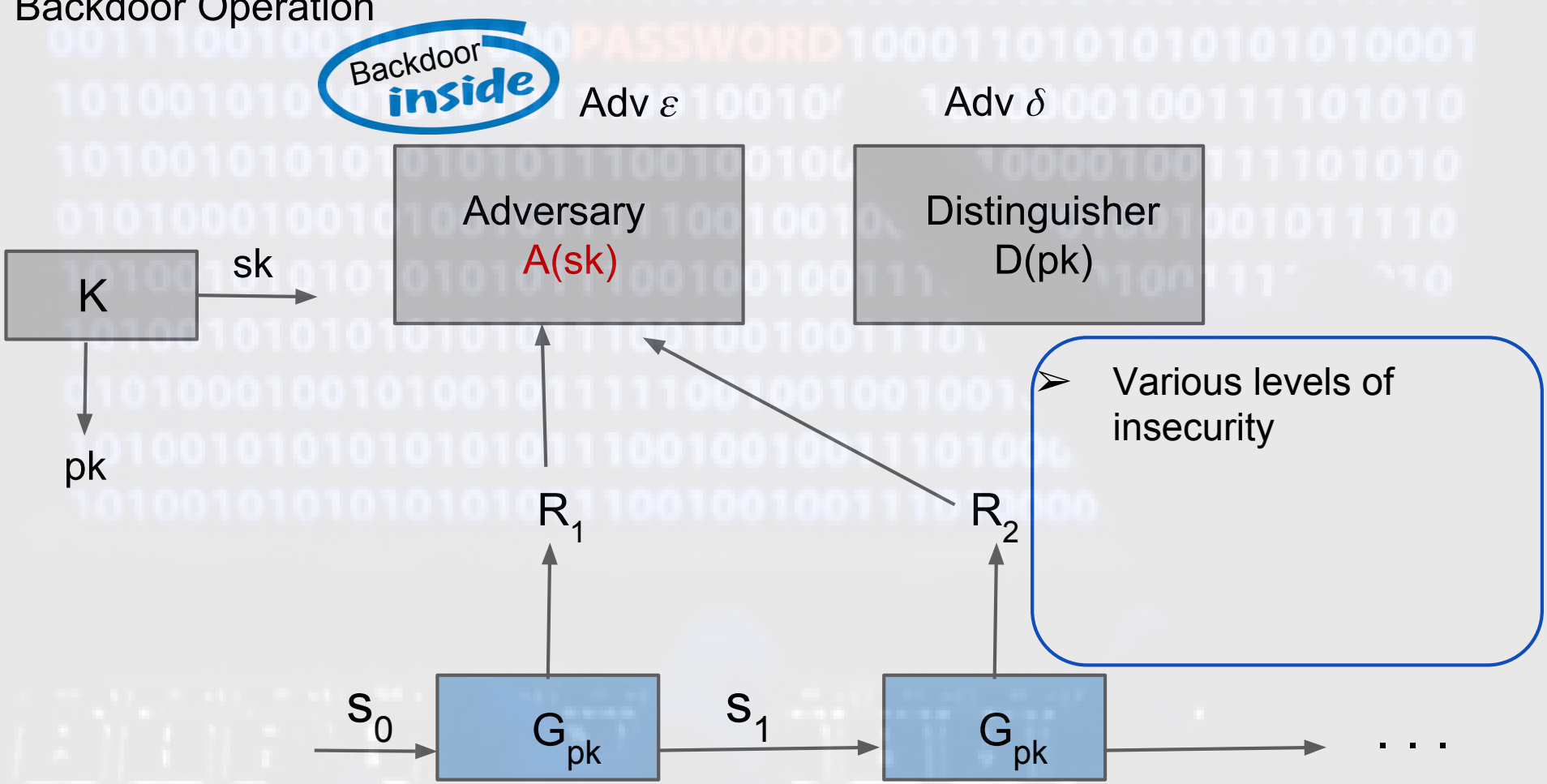
Backdoor Operation



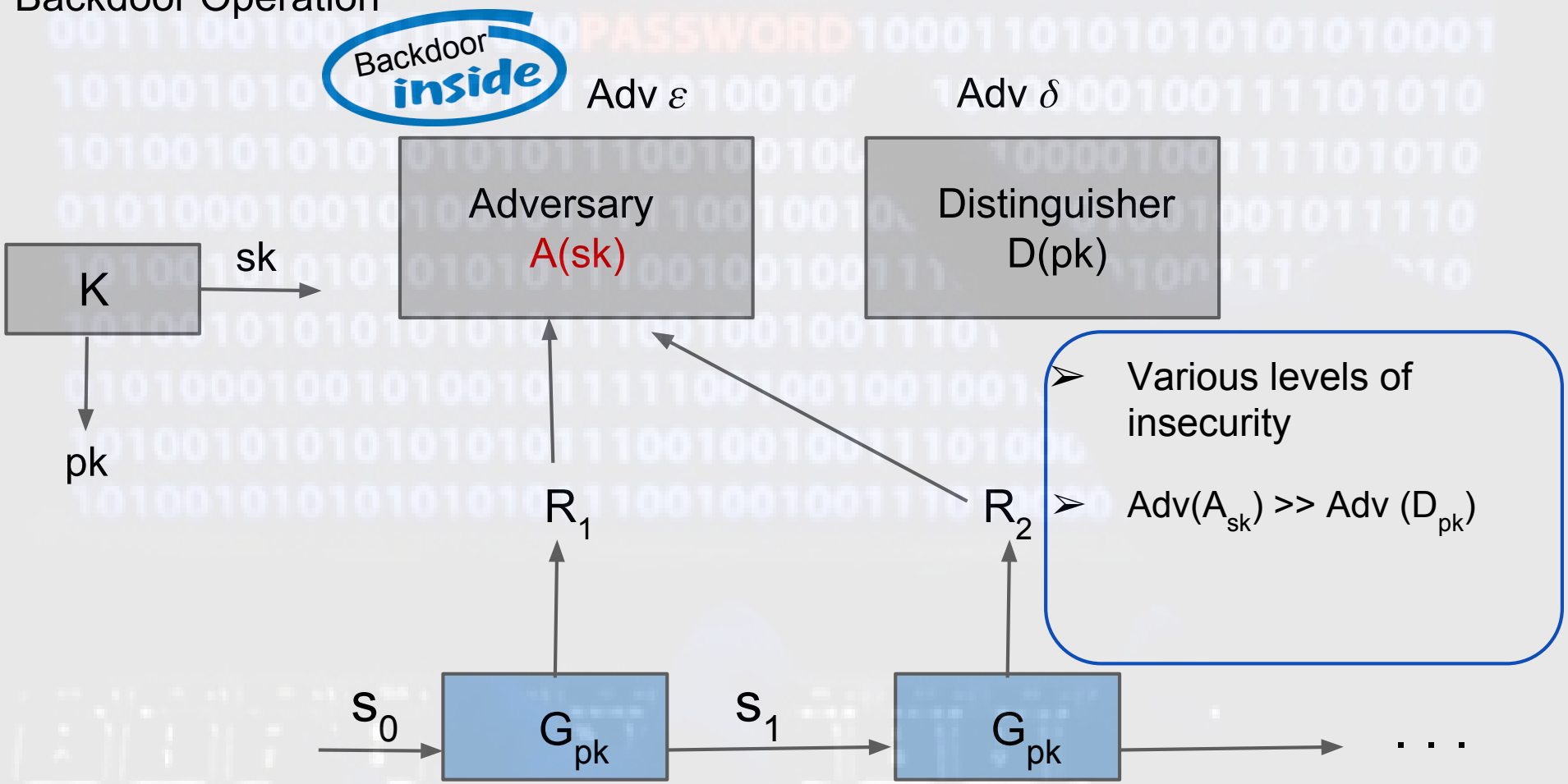
Backdoor Operation



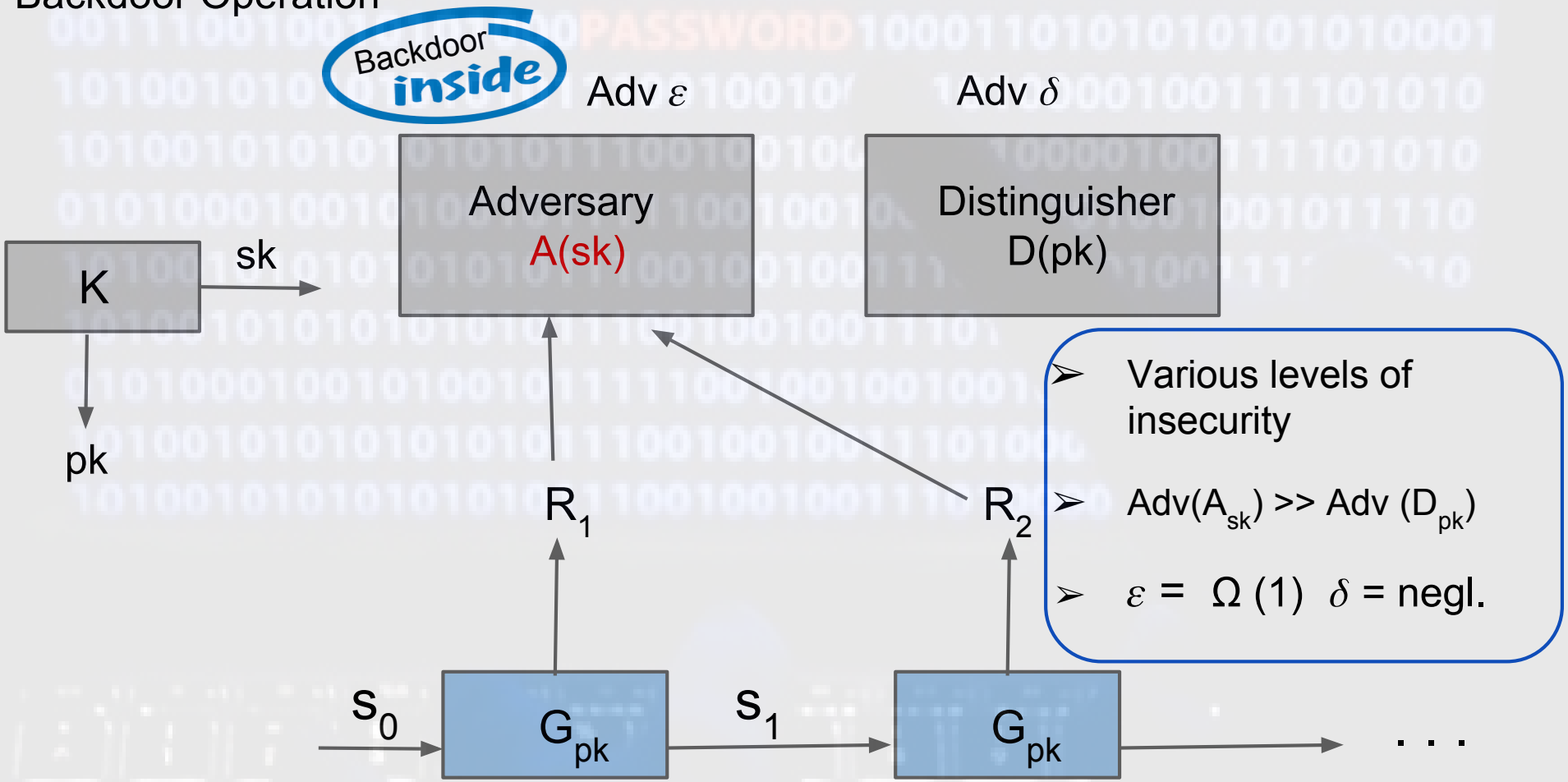
Backdoor Operation



Backdoor Operation

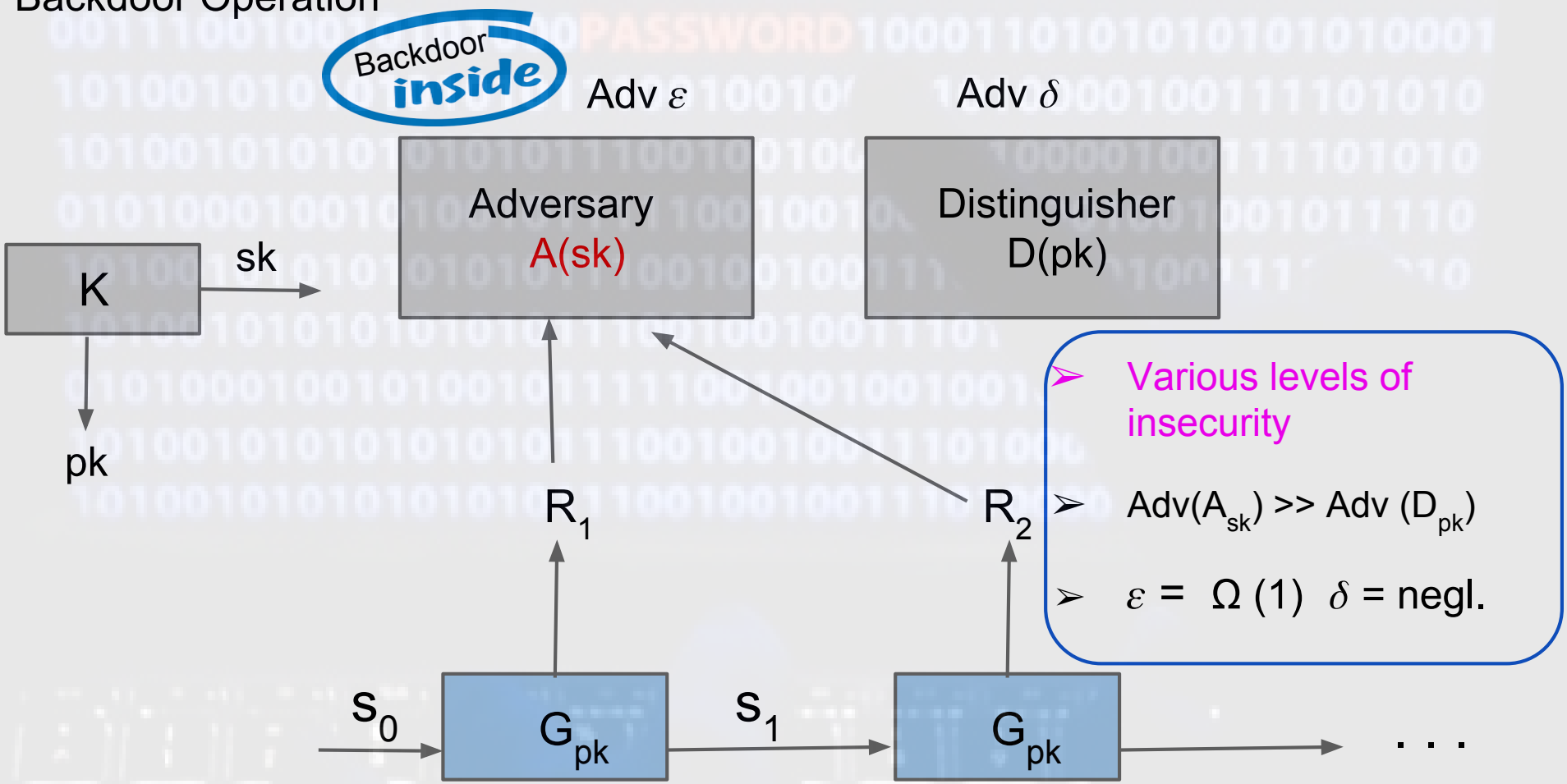


Backdoor Operation



- Various levels of insecurity
- $Adv(A_{sk}) >> Adv(D_{pk})$
- $\varepsilon = \Omega(1)$ δ = negl.

Backdoor Operation

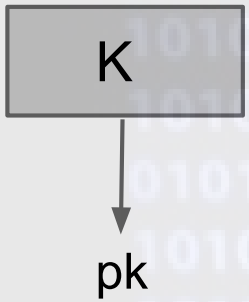


- Various levels of insecurity
- $Adv(A_{sk}) >> Adv(D_{pk})$
- $\varepsilon = \Omega(1)$ δ = negl.

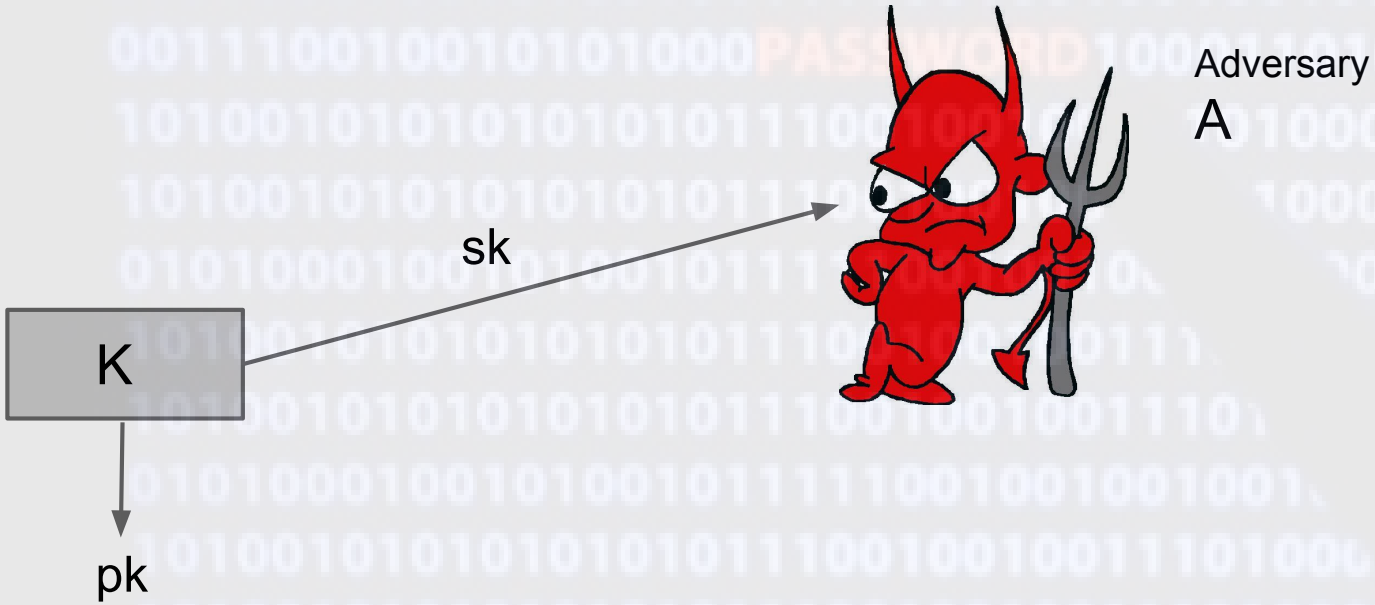
Distinguishing - $\mathcal{G}_{\text{dist}}$

Adversary

A



Distinguishing - $\mathcal{G}_{\text{dist}}$



Distinguishing - $\mathcal{G}_{\text{dist}}$

Adversary
A

sk

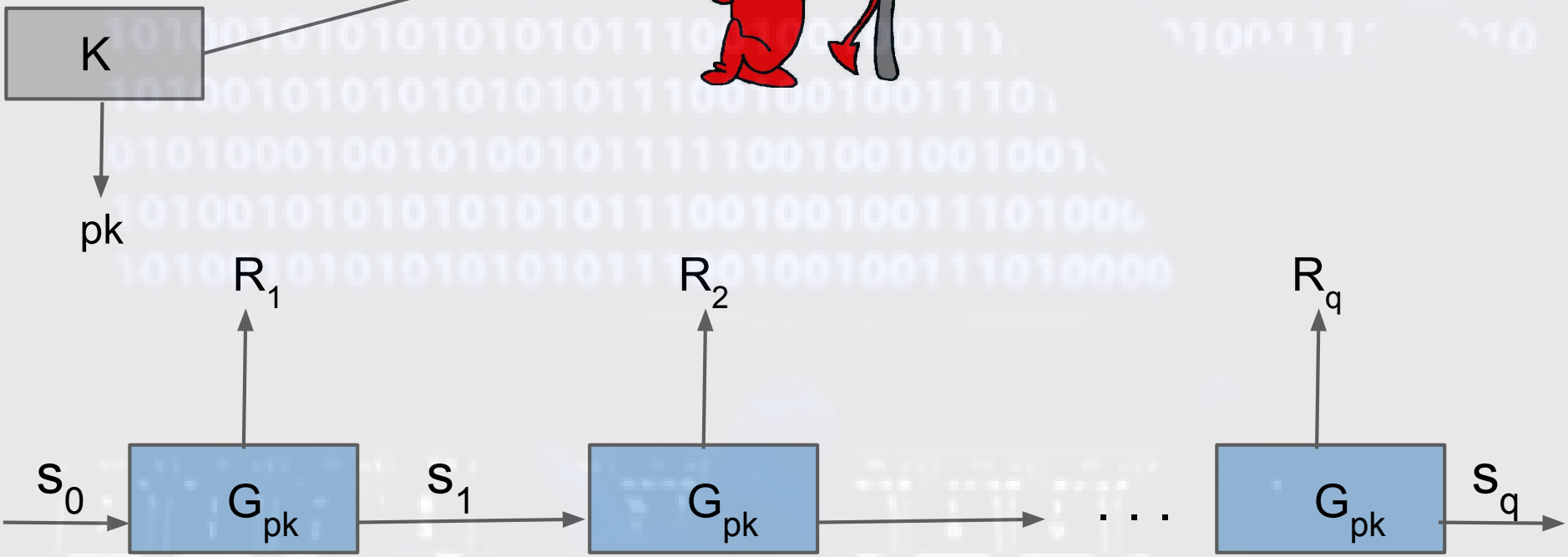
K

pk

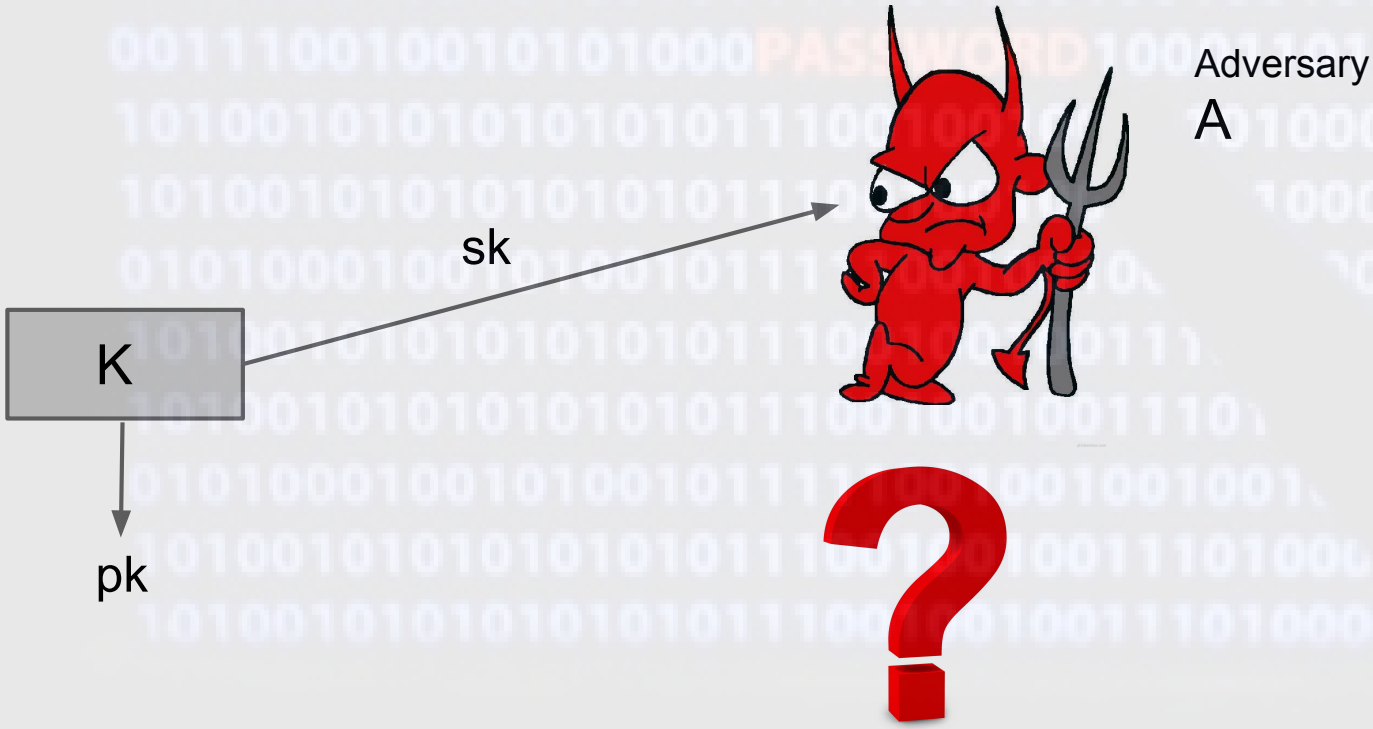
$\mathcal{U}_1$ $\mathcal{U}_2$ $\mathcal{U}_q$

Distinguishing - $\mathcal{G}_{\text{dist}}$

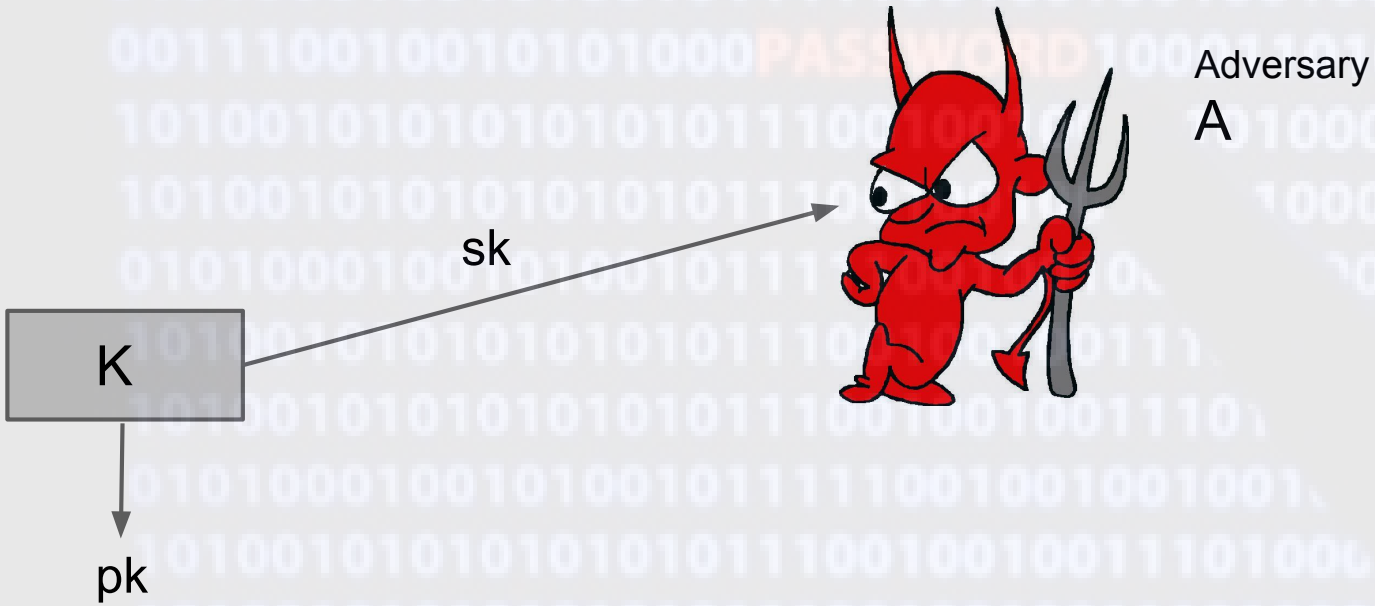
Adversary
A



Distinguishing - $\mathcal{G}_{\text{dist}}$

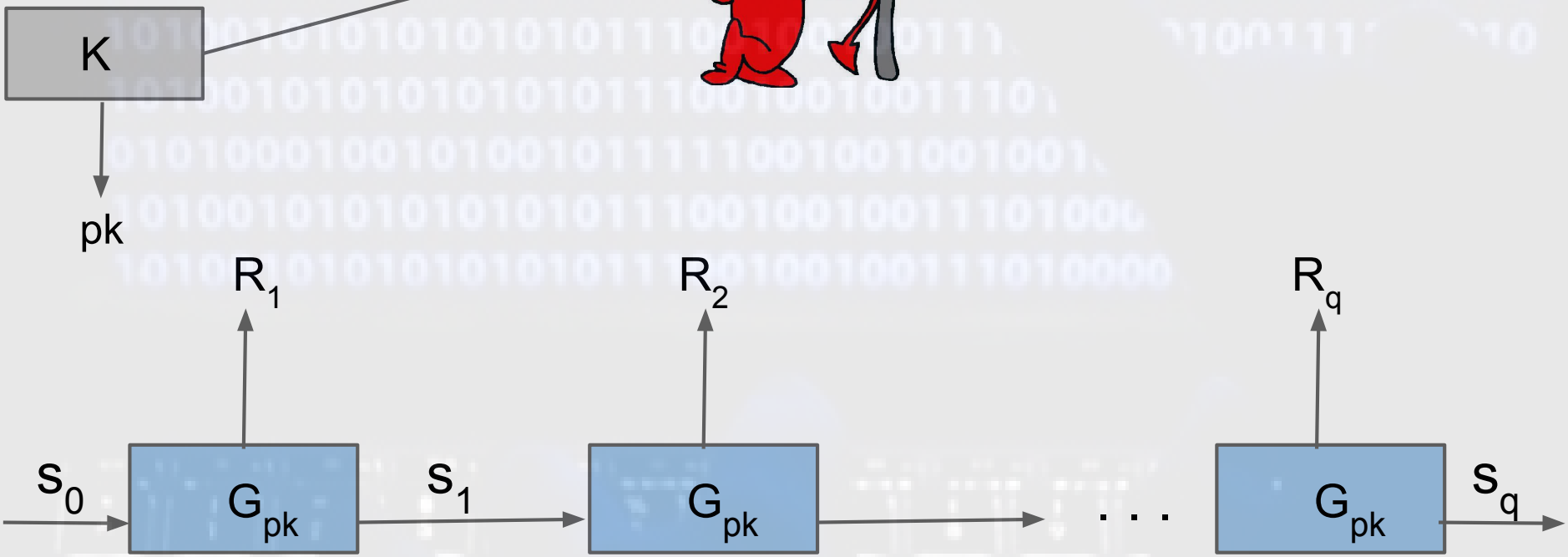


Next state prediction - $\mathcal{G}_{\text{next}}$

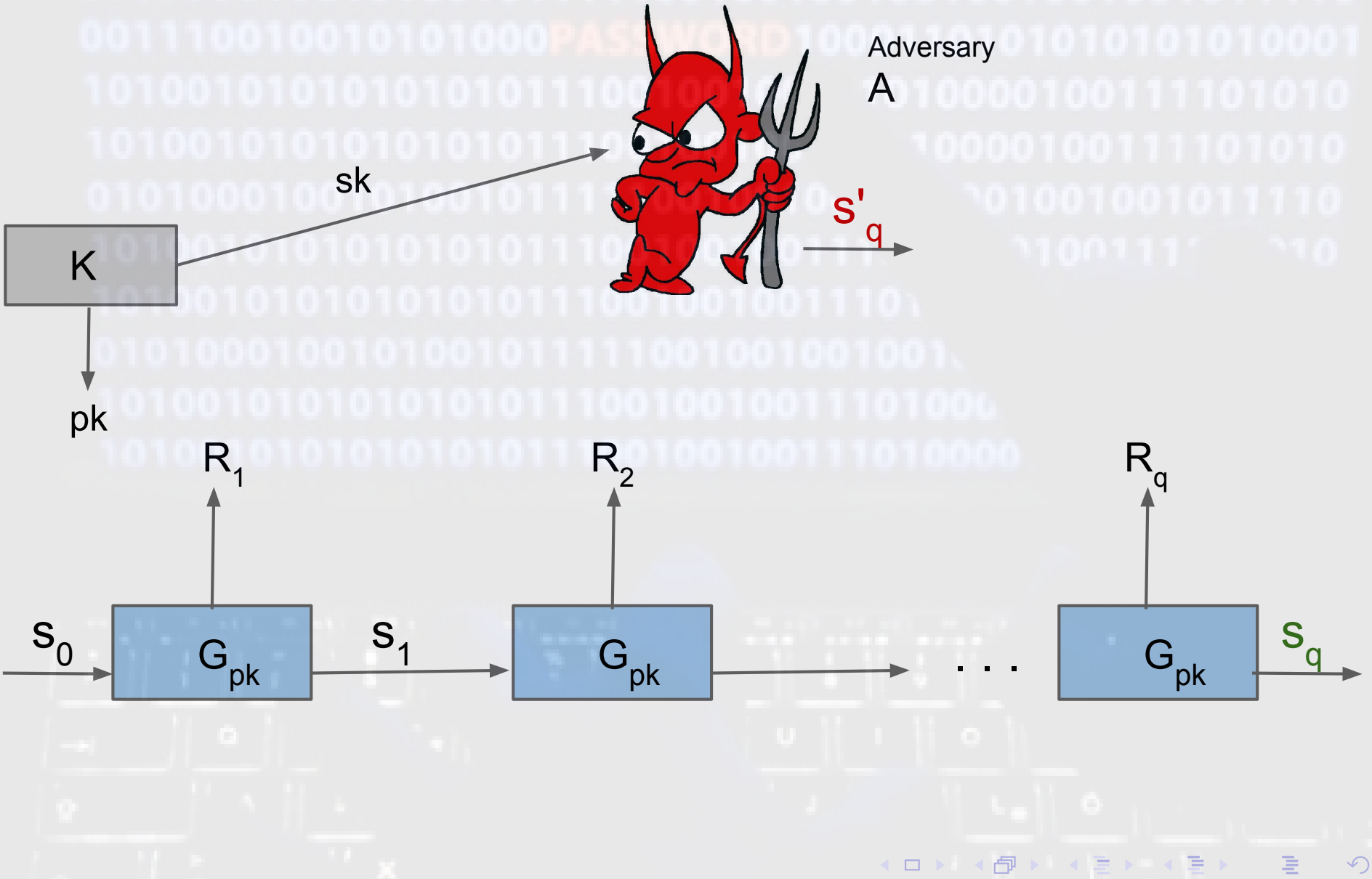


Next state prediction - $\mathcal{G}_{\text{next}}$

Adversary
A



Next state prediction - $\mathcal{G}_{\text{next}}$



Next state prediction - $\mathcal{G}_{next}$

Adversary
A

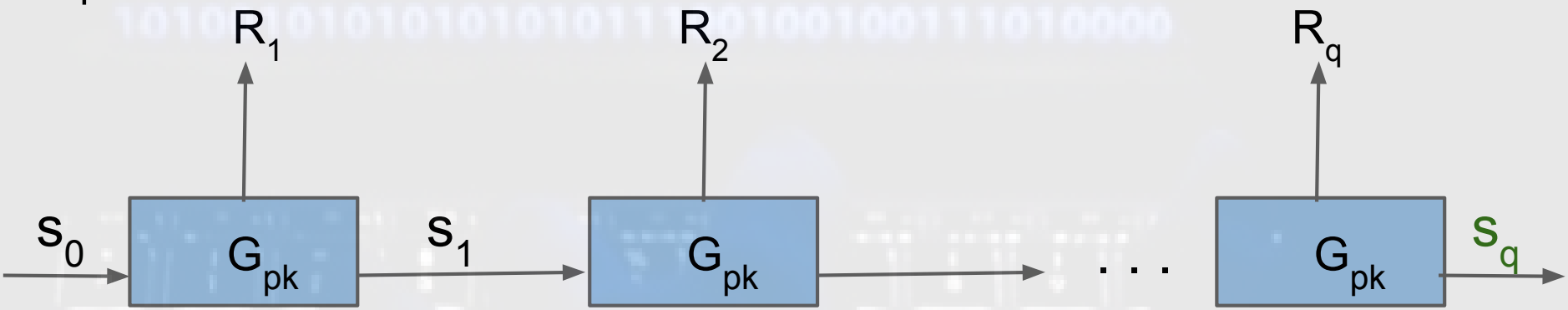
sk

K

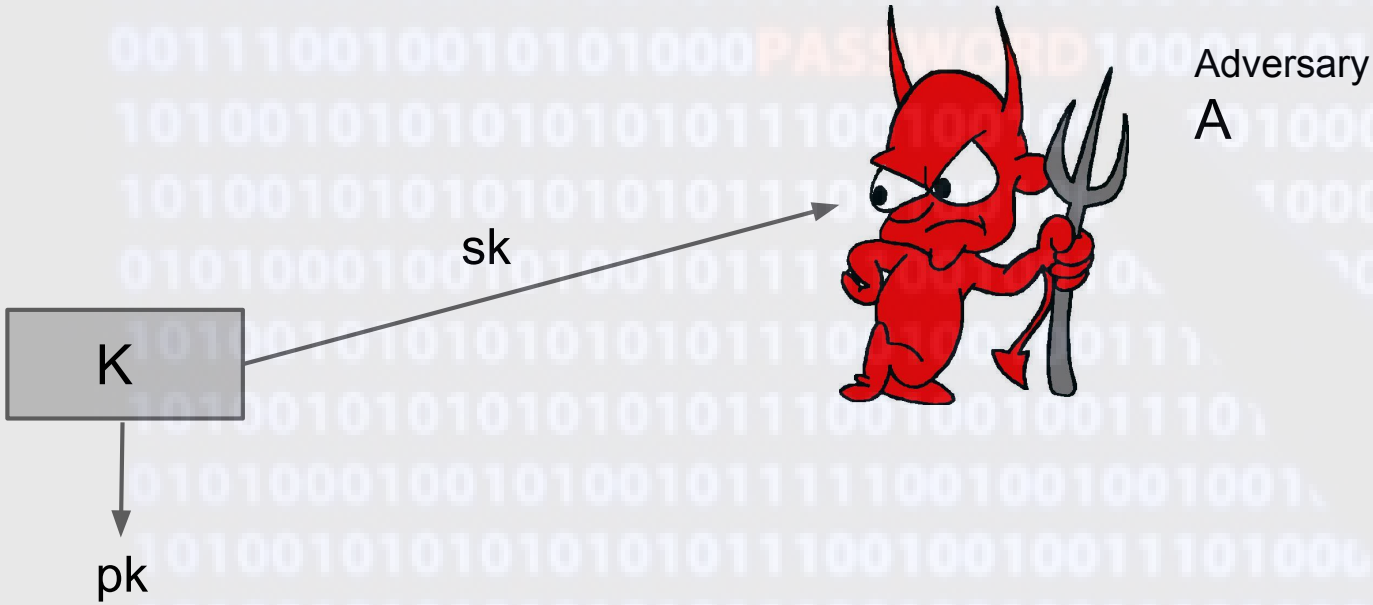
pk

s'_q

$$Adv\ (A_{sk}) = Pr\ [\ s'_q = s_q\ ]$$



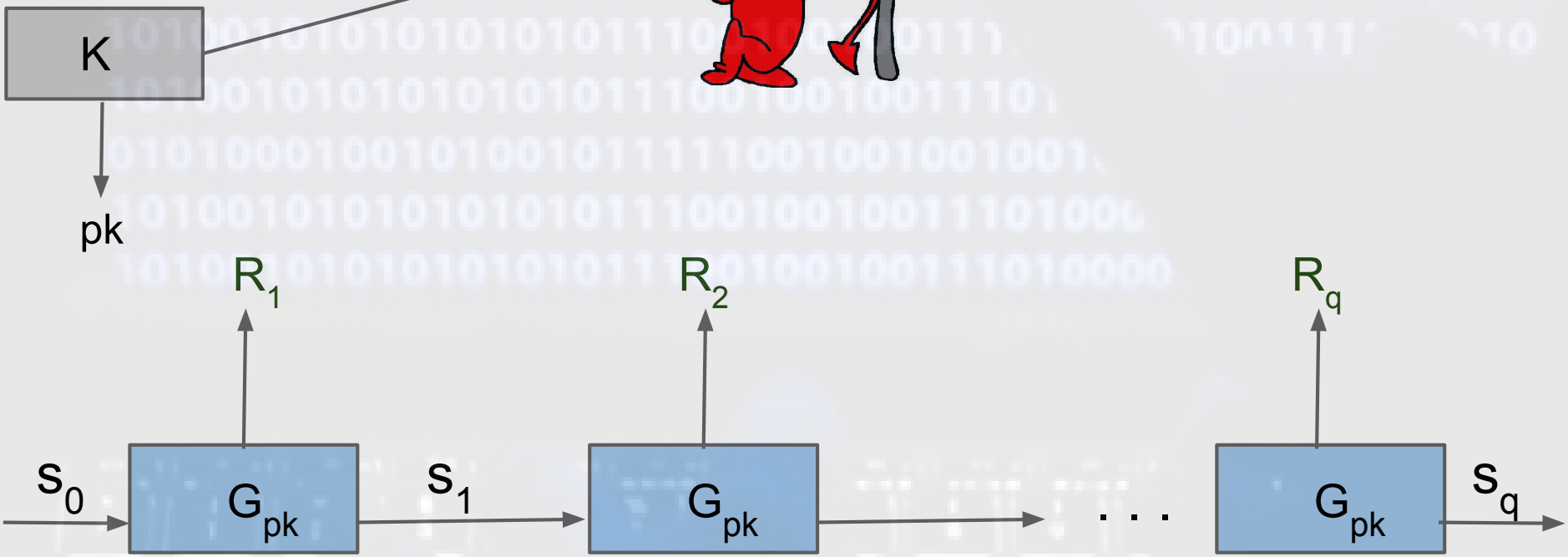
Random seek - $\mathcal{G}_{\text{rseek}}$



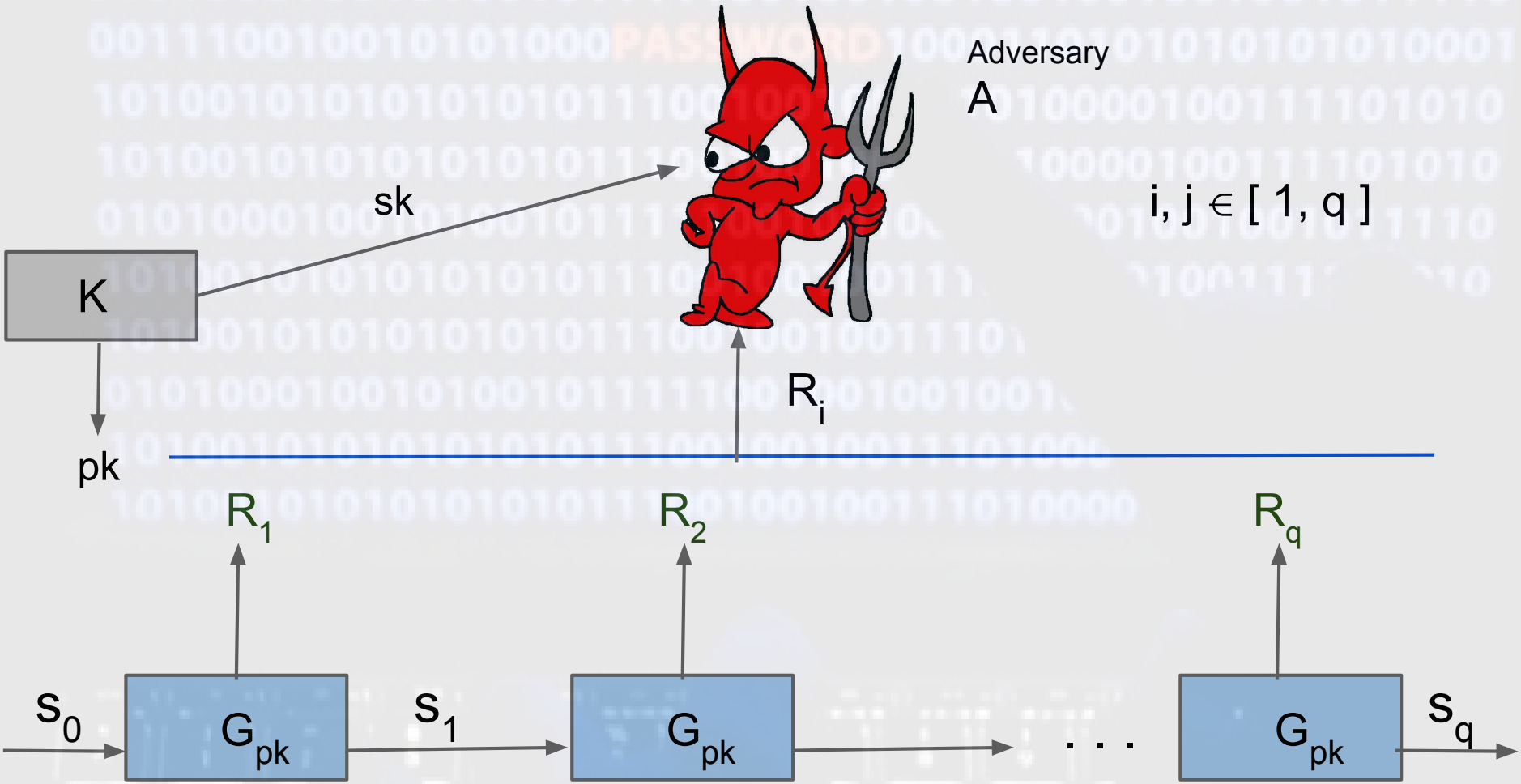
Random seek - $\mathcal{G}_{\text{rseek}}$

Adversary
A

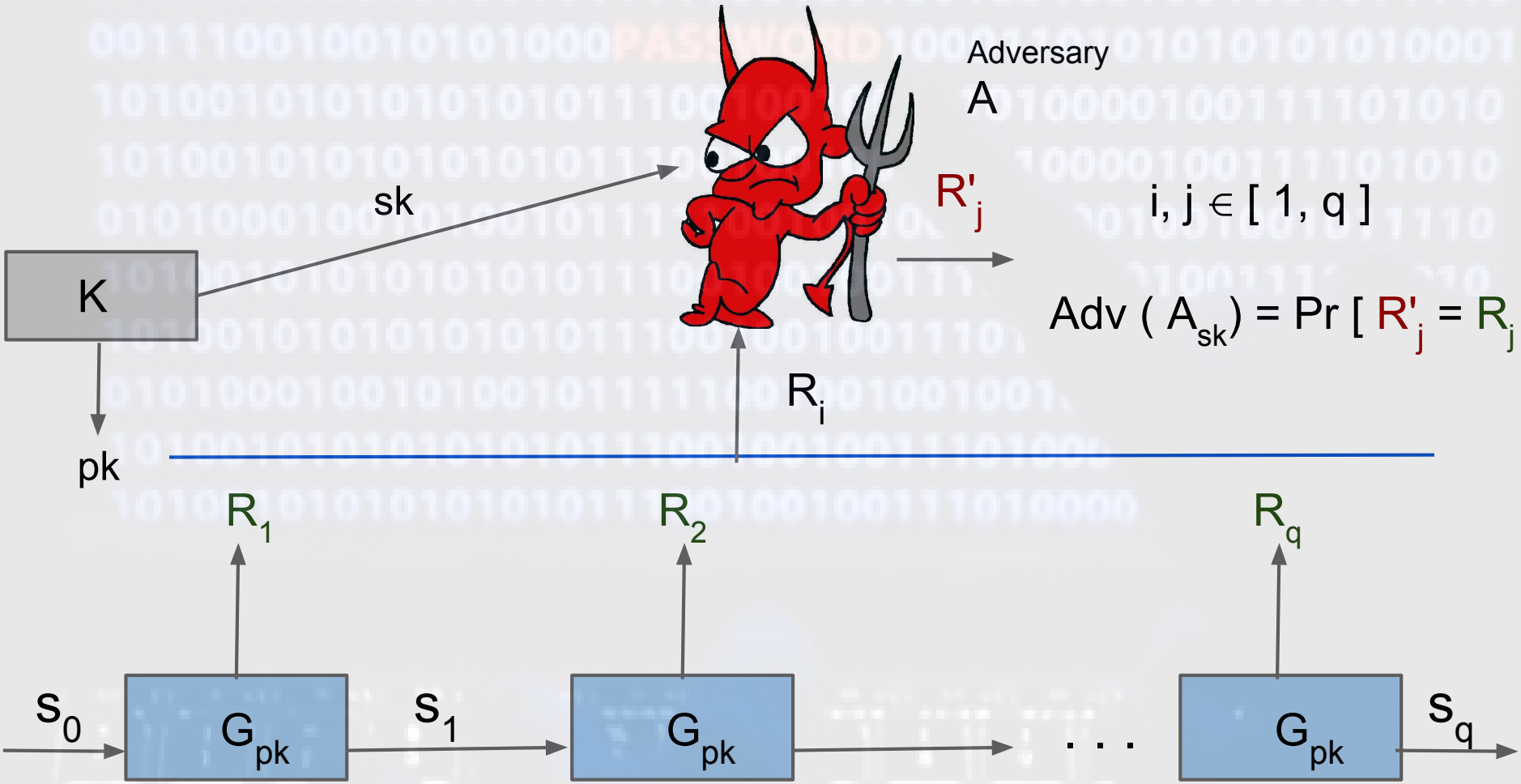
$i, j \in [\,1, q\,]$



Random seek - $\mathcal{G}_{\text{rseek}}$



Random seek - $\mathcal{G}_{\text{rseek}}$



Summary of results

- Definitional framework of Backdoored PRGs.
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts

$$\mathcal{G}_{\text{dist}} \implies \textit{IND\$-CPA} \implies \mathcal{G}_{\text{next}}, \mathcal{G}_{\text{rseek}}$$

- Investigate countermeasures to BPRGs - immunizers.
  - (In)effectiveness of countermeasures
  - Provably secure solution

Summary of results

- Definitional framework of Backdoored PRGs.
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts

$$\mathcal{G}_{\mathsf{dist}} \implies \boxed{\mathit{IND\$\text{-}CPA} \implies \mathcal{G}_{\mathsf{next}}, \mathcal{G}_{\mathsf{rseek}}}$$

- Investigate countermeasures to BPRGs - immunizers.
  - (In)effectiveness of countermeasures
  - Provably secure solution

Key Encapsulation Mechanism

A KEM scheme is a triple of algorithms $(\text{KeyGen}, \text{Encap}, \text{Decap})$.

- The KeyGen outputs a public/secret key pair,

$$(pk, sk) \leftarrow \text{KeyGen}$$

- The encapsulation algorithm

$$(c, K) \leftarrow \text{Encap}(pk; r), K \in \{0,1\}^n$$

- The decapsulation algorithm

$$\text{Decap}(sk, c) = \tilde{K} \in \{0,1\}^n \cup \{\text{invalid}\}$$

KEM

- **Correctness**: With all but negl. probability,

$$\mathrm{Decap}(sk, c) = K \text{ for } (c, K) = \mathrm{Encap}(pk; r)$$

- **Security**: The outputs of Encap indistinguishable from a pair of random bit strings.
  - Ciphertext pseudorandomness - stronger than usual KEM notion.

$\mathcal{G}_{\text{next}}$-BPRG from KEM

Γ = (Gen, Encap, Decap) a pseudorandom-ciphertext KEM

$\mathcal{G}_{\text{next}}$-BPRG from KEM

Γ = (Gen, Encap, Decap) a pseudorandom-ciphertext KEM

K :

(pk, sk) $\leftarrow$ Gen
return (pk,sk)

$\mathcal{G}_{\text{next}}$-BPRG from KEM

Γ = (Gen, Encap, Decap) a pseudorandom-ciphertext KEM

K :

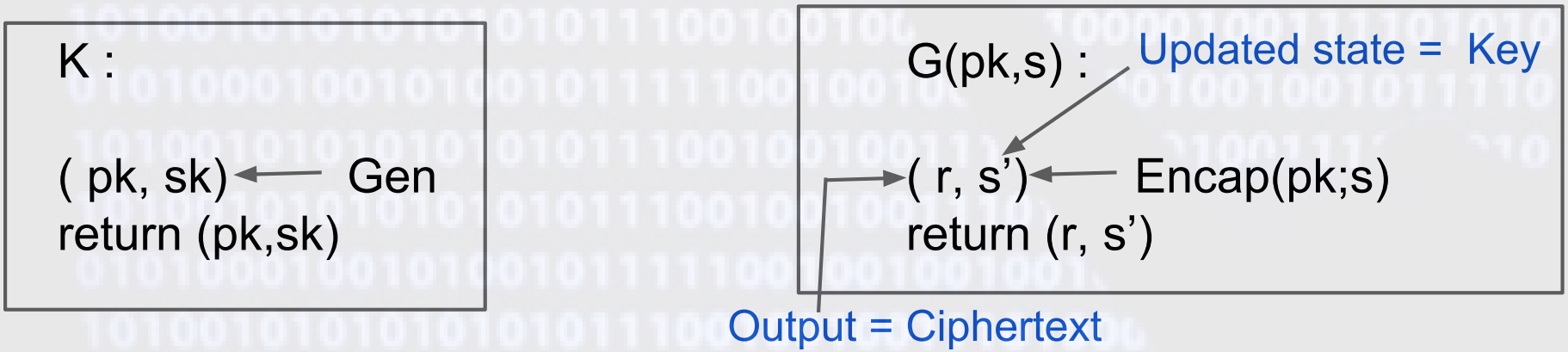
(pk, sk) ← Gen
return (pk,sk)

G(pk,s) :

(r, s') ← Encap(pk;s)
return (r, s')

$\mathcal{G}_{\text{next}}$-BPRG from KEM

Γ = (Gen, Encap, Decap) a pseudorandom-ciphertext KEM



$\mathcal{G}_{\text{next}}$-BPRG from KEM

Γ = (Gen, Encap, Decap) a pseudorandom-ciphertext KEM

K :

(pk, sk) ← Gen
return (pk,sk)

G(pk,s) :

Updated state = Key

(r, s') ← Encap(pk;s)
return (r, s')

Output = Ciphertext

A (sk, r_1, … , r_q) :

s' ← Decap(sk, r_q)
return s'

$\mathcal{G}_{next}$-BPRG from KEM

Γ = (Gen, Encap, Decap) a pseudorandom-ciphertext KEM

K :

(pk, sk) ← Gen
return (pk,sk)

G(pk,s) : Updated state = Key

(r, s') ← Encap(pk;s)
return (r, s')

Output = Ciphertext

A (sk, r_1, … , r_q) :

s' ← Decap(sk, r_q)
return s'

- Attack ← correctness of KEM
- Standard PRG security ← ciphertext pseudorandomness

Summary of results

- Definitional framework of Backdoored PRGs.
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts

$$\mathcal{G}_{\text{dist}} \implies \textit{IND\$-CPA} \implies \mathcal{G}_{\text{next}}, \mathcal{G}_{\text{rseek}}$$

- Investigate countermeasures to BPRGs - immunizers.
  - (In)effectiveness of countermeasures
  - Provably secure solution

Summary of results

- Definitional framework of Backdoored PRGs.
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts

$$\boxed{\mathcal{G}_{\mathsf{dist}} \implies \mathit{IND\$\text{-}CPA}} \implies \mathcal{G}_{\mathsf{next}}, \mathcal{G}_{\mathsf{rseek}}$$

- Investigate countermeasures to BPRGs - immunizers.
  - (In)effectiveness of countermeasures
  - Provably secure solution

Public Key Encryption from BPRG

- We show that the existence of BPRGs implies public-key encryption (PKE).
- From a backdoored PRG, we construct a bit encryption scheme with noticeable correctness and overwhelming secrecy.
- *Amplify* - Parallel repetition and privacy amplification of key-agreement (Holenstein 2005), amplify secrecy and correctness without increasing the number of rounds.
- Since the number of rounds is not increased, we obtain secure public-key encryption.

Public Key Encryption from $\mathcal{G}_{\mathsf{dist}}$-BPRG

- KeyGen: $(pk, sk) \leftarrow \mathrm{K}$

Public Key Encryption from $\mathcal{G}_{\text{dist}}$-BPRG

- KeyGen: $(pk, sk) \leftarrow \mathrm{K}$
- Encryption:

Public Key Encryption from $\mathcal{G}_{\text{dist}}$-BPRG

- KeyGen: $(pk, sk) \leftarrow \mathrm{K}$
- Encryption:
  - To encrypt bit 0, ciphertext is set to uniformly random string

Public Key Encryption from $\mathcal{G}_{\text{dist}}$-BPRG

- KeyGen: $(pk, sk) \leftarrow \mathrm{K}$
- Encryption:
  - To encrypt bit 0, ciphertext is set to uniformly random string
  - To encrypt bit 1, ciphertext is the output of $\mathcal{G}_{\text{dist}}$-BPRG

Public Key Encryption from $\mathcal{G}_{\text{dist}}$-BPRG

- KeyGen: $(pk, sk) \leftarrow \mathrm{K}$
- Encryption:
  - To encrypt bit 0, ciphertext is set to uniformly random string
  - To encrypt bit 1, ciphertext is the output of $\mathcal{G}_{\text{dist}}$-BPRG
- Decryption: Call A with the secret key as trapdoor

Public Key Encryption from $\mathcal{G}_{\text{dist}}$-BPRG

- KeyGen: $(pk, sk) \leftarrow \mathrm{K}$
- Encryption:
  - To encrypt bit 0, ciphertext is set to uniformly random string
  - To encrypt bit 1, ciphertext is the output of $\mathcal{G}_{\text{dist}}$-BPRG
- Decryption: Call A with the secret key as trapdoor
- Correctness of decryption - by advantage of A in the $\mathcal{G}_{\text{dist}}$ game

Public Key Encryption from $\mathcal{G}_{\text{dist}}$-BPRG

- KeyGen: $(pk, sk) \leftarrow \mathrm{K}$
- Encryption:
  - To encrypt bit 0, ciphertext is set to uniformly random string
  - To encrypt bit 1, ciphertext is the output of $\mathcal{G}_{\text{dist}}$-BPRG
- Decryption: Call A with the secret key as trapdoor
- Correctness of decryption - by advantage of A in the $\mathcal{G}_{\text{dist}}$ game
- Security - by standard PRG security for distinguishers without the trapdoor.

Public Key Encryption from BPRG

- Backdoored PRG constructions from KEM (equivalent to PKE)
- Public key encryption from a backdoored PRG.

Theorem (Informal)

Backdoor PRGs exist iff public-key encryption with pseudorandom ciphertexts exists.

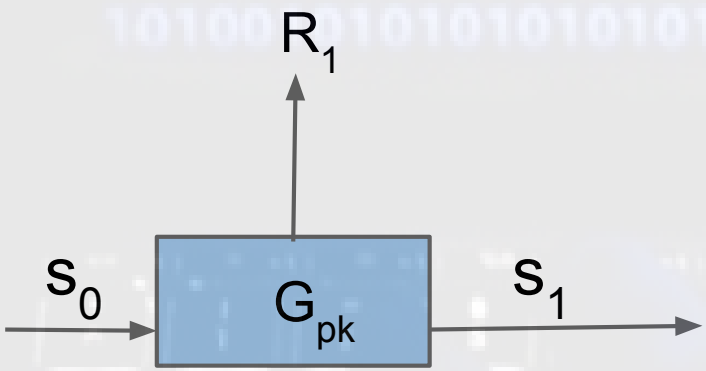
Summary of results

- Definitional framework of Backdoored PRGs.
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts

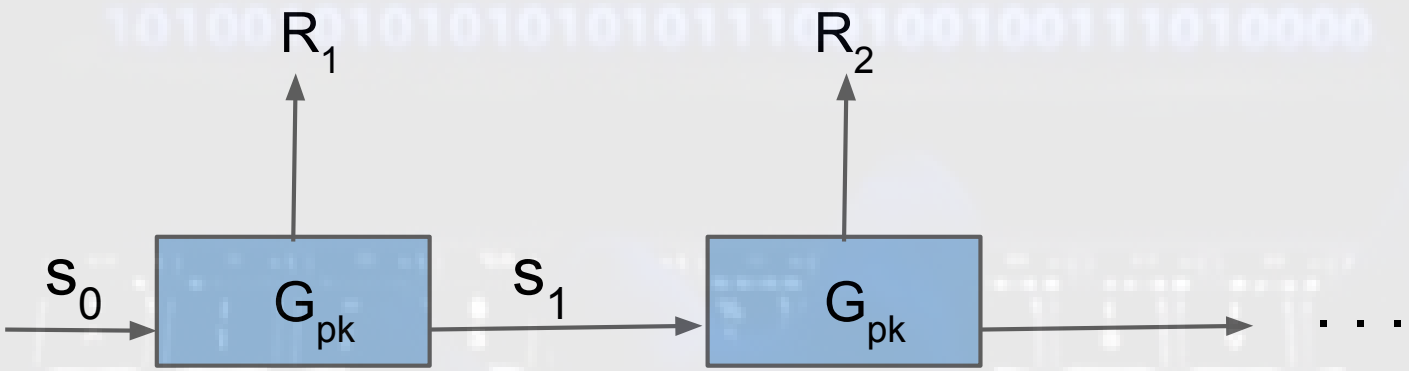
$$\mathcal{G}_{\text{dist}} \implies \mathit{IND\$\text{-}CPA} \implies \mathcal{G}_{\text{next}}, \mathcal{G}_{\text{rseek}}$$

- Investigate countermeasures to BPRGs - immunizers
  - (In)effectiveness of countermeasures
  - Provably secure solution

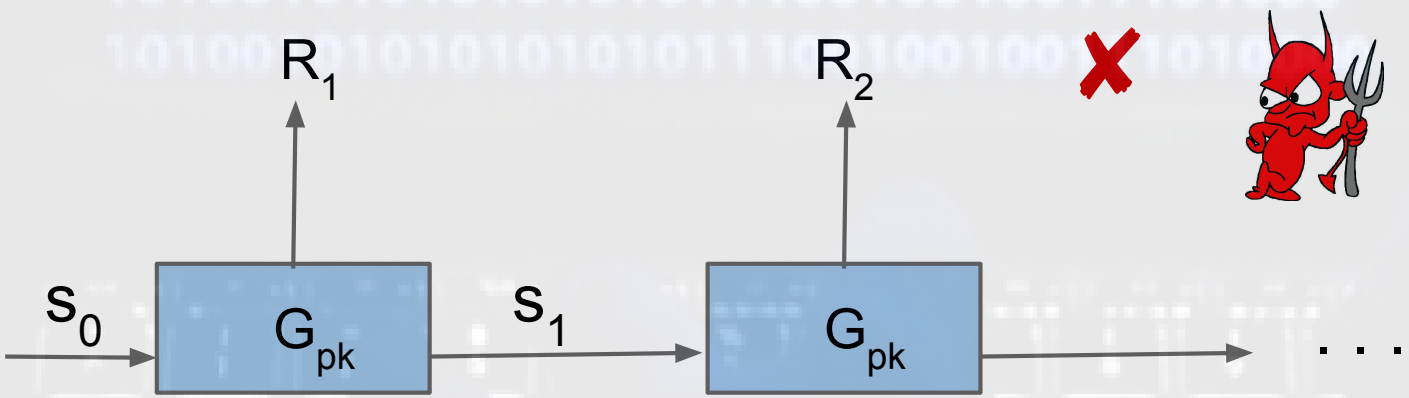
Immunization



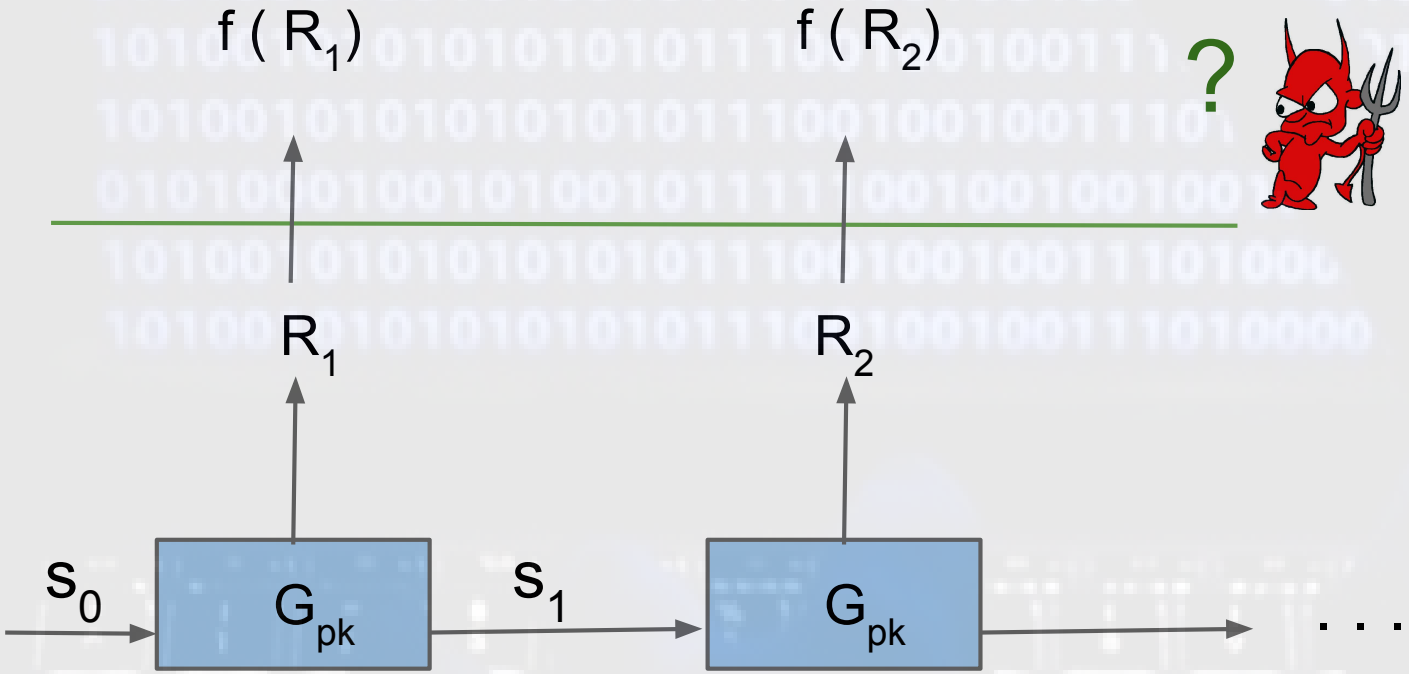
Immunization



Immunization

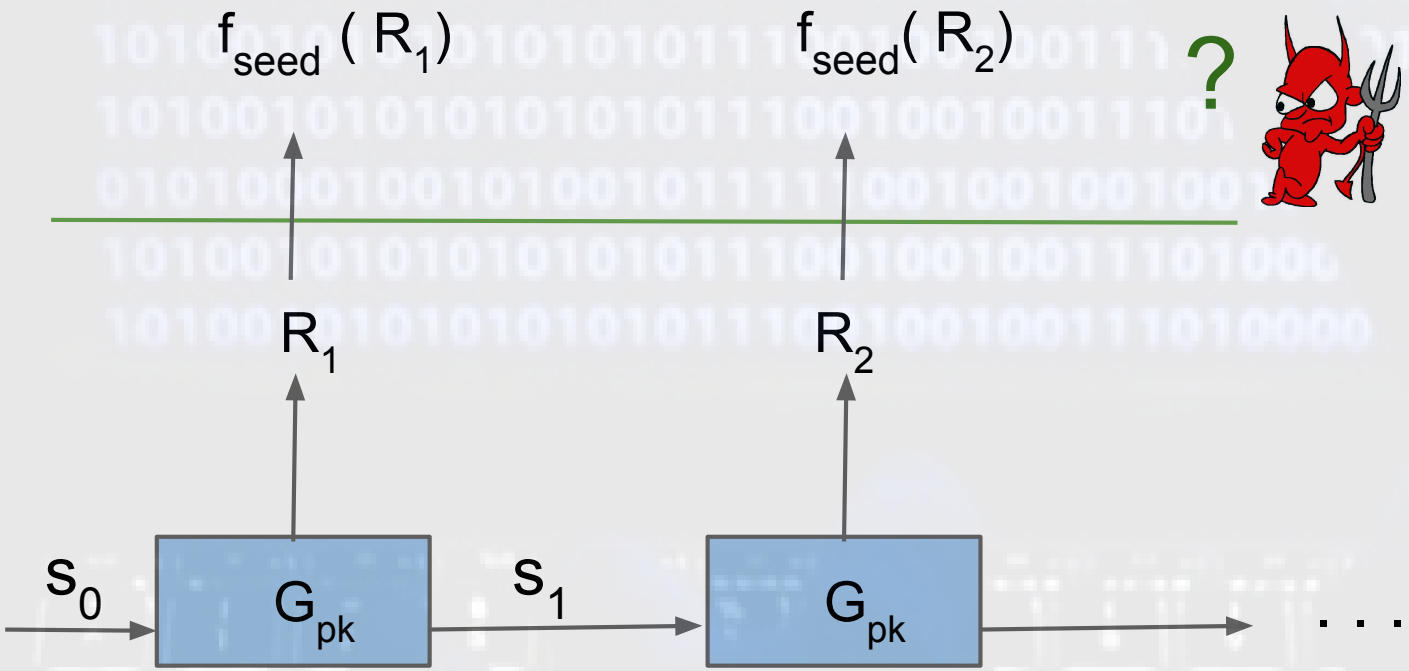


Immunization



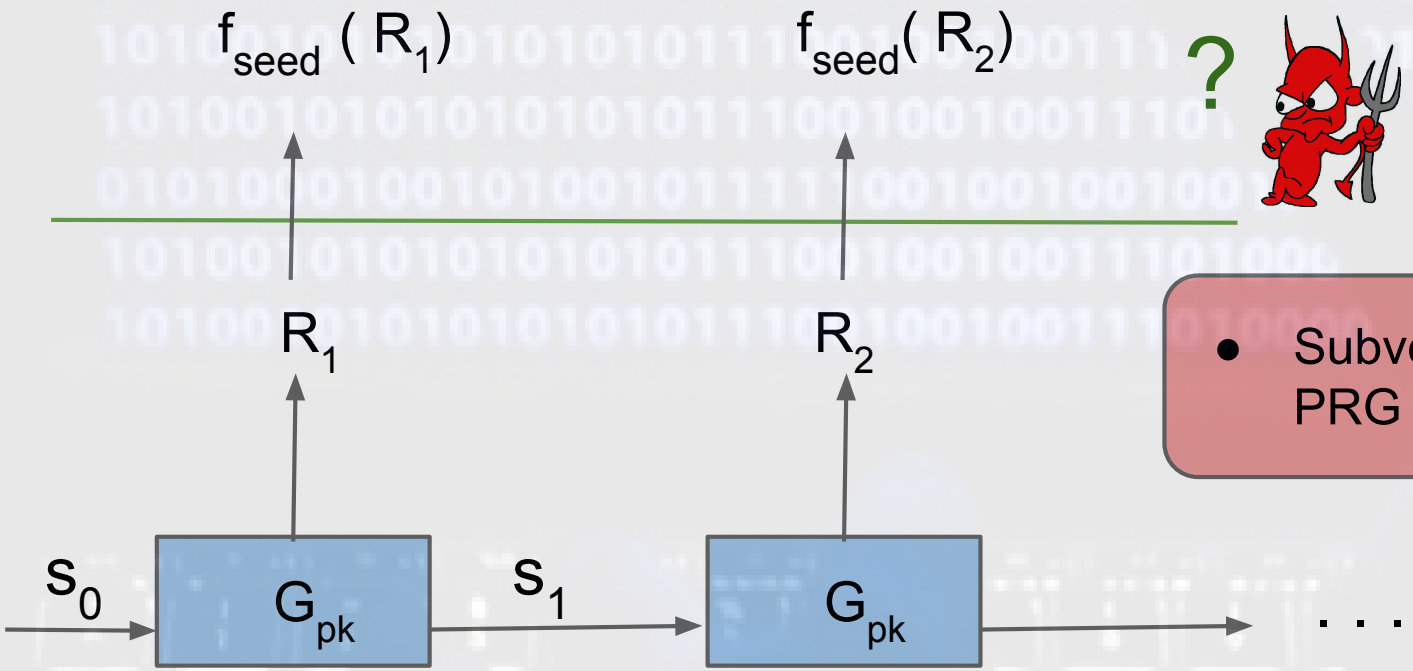
Immunization

Family of functions $\{ f_{seed} \mid seed \in \{0,1\}^k \}$ seed $\leftarrow$ uniformly random



Immunization

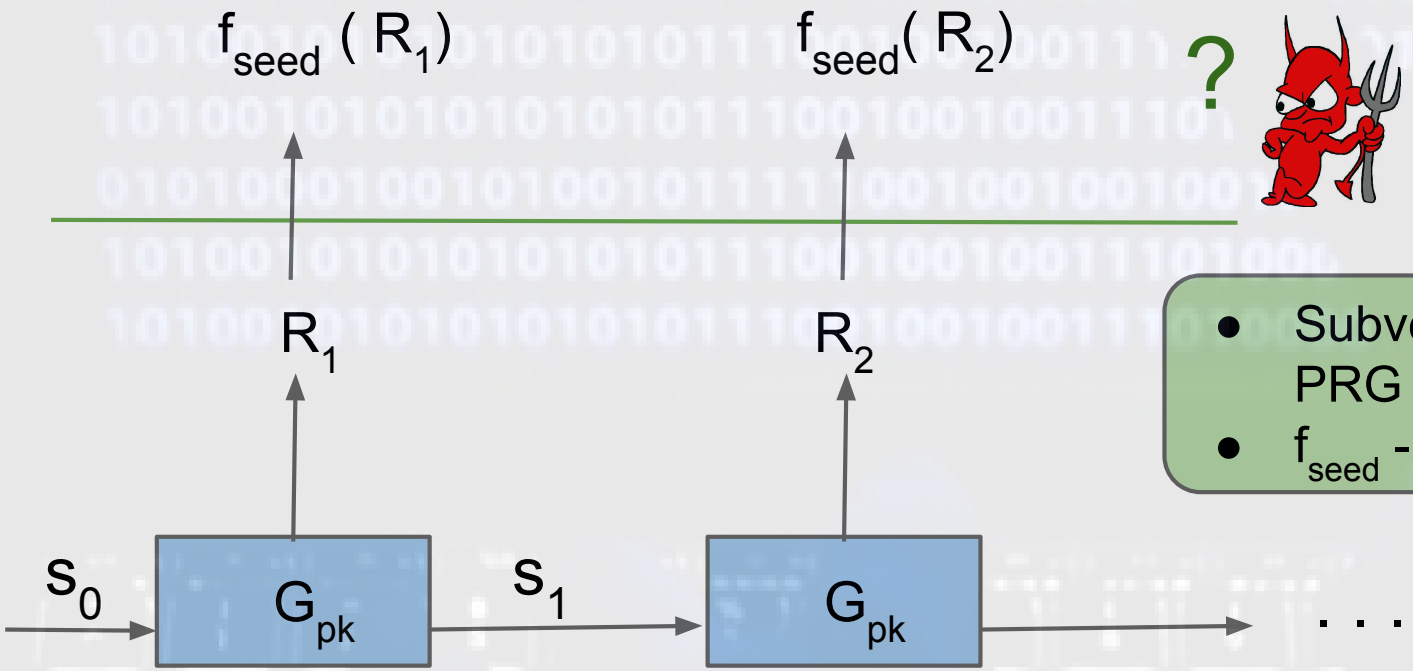
Family of functions $\{ f_{seed} \mid seed \in \{0,1\}^k \}$ seed $\leftarrow$ uniformly random



- Subverted standard PRG

Immunization

Family of functions $\{ f_{seed} \mid seed \in \{0,1\}^k \}$ $seed \leftarrow$ uniformly random



- Subverted standard PRG
- f_{seed} - User's choice

Immunization models

- **Public** immunization: Both G and A know seed.
  - seed is revealed to the attacker A prior to construction of G.
- **Semi-private** immunization: A knows seed, G does not.
  - G is constructed without reference to seed. The attacker A learns seed, and thus f_{seed}, only *after* the specification of G
- **Private** immunization: seed is secret from both A and G.
  - G is constructed without reference to seed and A *never* learns seed.

Results in Immunization models

- Negative result in the public model - BPRG against any immunization family
- (Non-trivial) Positive results in the semi-private model
- (Trivial) Positive and (initial) negative results in the private model

Immunization models

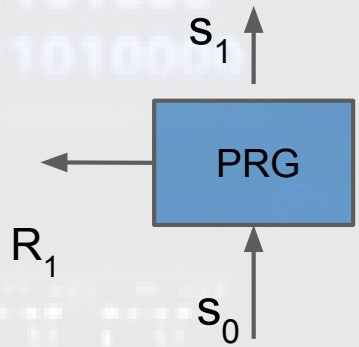
- Public immunization
- Semi-private immunization
- Private immunization

Public randomness

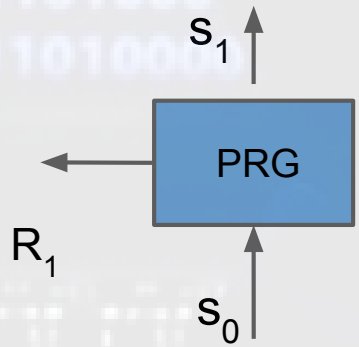
Key idea:

- Prepare a string c that is pseudorandom without sk
- c gives away some information with the knowledge of sk
- "Leak" c bit-by-bit through the PRG outputs
  - Skip outputs until $[f(.)]_1$ is the bit to be leaked - rejection sampling
  - Leakage undetectable to user as c is pseudorandom without sk

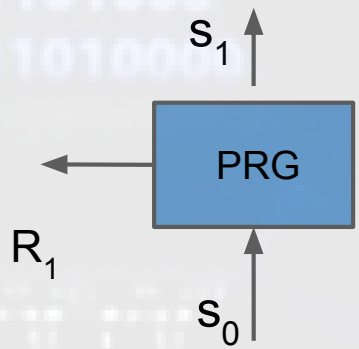
c - pseudorandom string



c - pseudorandom string
Leak a bit of c

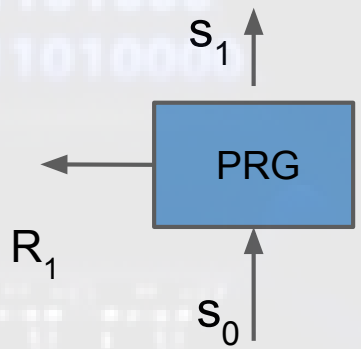


$$[\, f_{seed}(R_1) \,]_1 = c_j \; ?$$

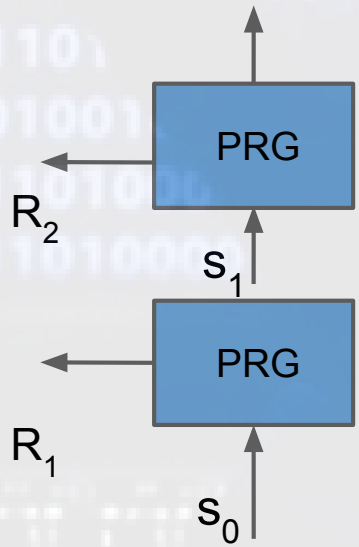


NO

$$[\, f_{seed}(R_1) \,]_1 = c_j \, ?$$

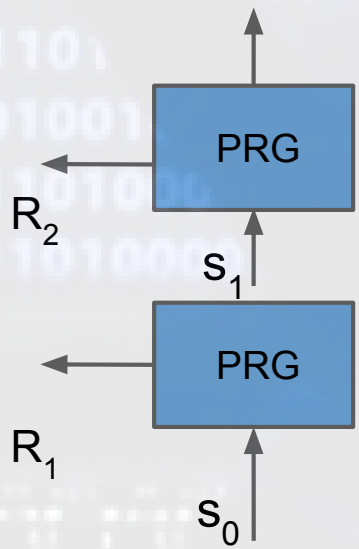


$$[\, f_{seed}(R_2) \,]_1 = c_j \; ?$$



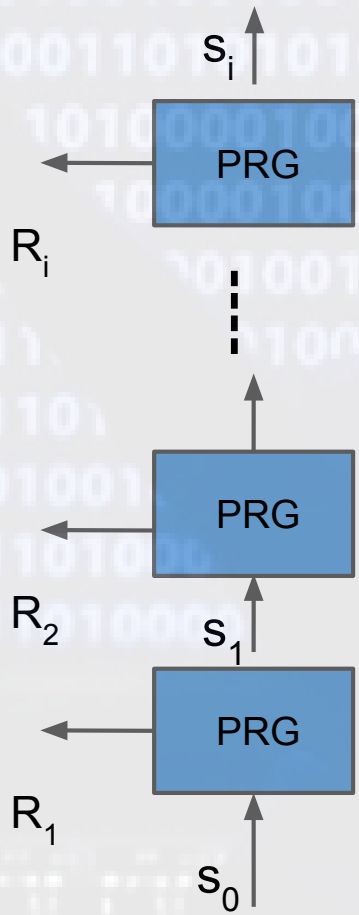
NO

$$[\, f_{seed}(\, R_2 \,)\,]_1 = c_j \,?$$



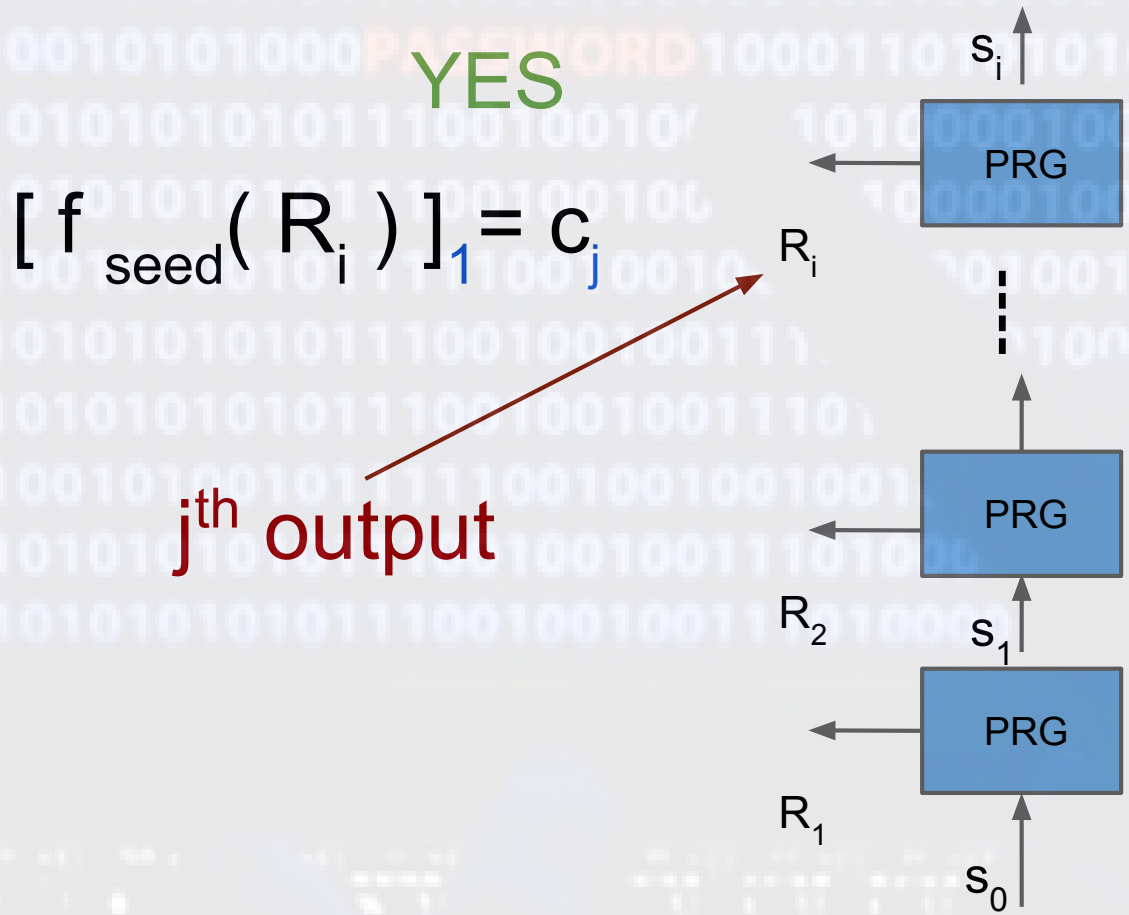
YES

$$[\, f_{seed}(\, R_i \,)\,]_1 = c_j \; ?$$



YES

$$[\, f_{seed}(\, R_i \,) \,]_1 = c_j$$



j^{th} output

Public randomness

The high-level construction:

1. BPRG in two phases - *leakage* phase and *normal* phase

Public randomness

The high-level construction:

1. BPRG in two phases - *leakage* phase and *normal* phase
2. Use the **key idea** in an initial *leakage* phase - leak something useful.

Public randomness

The high-level construction:

1. BPRG in two phases - *leakage* phase and *normal* phase
2. Use the **key idea** in an initial *leakage* phase - leak something useful.
   - Pseudorandom ciphertext encrypting a future state

Public randomness

The high-level construction:

1. BPRG in two phases - *leakage* phase and *normal* phase
2. Use the **key idea** in an initial *leakage* phase - leak something useful.
   - Pseudorandom ciphertext encrypting a future state
3. The trapdoor is the secret key of the PKE

Public randomness

The high-level construction:

1. BPRG in two phases - *leakage* phase and *normal* phase
2. Use the **key idea** in an initial *leakage* phase - leak something useful.
   - Pseudorandom ciphertext encrypting a future state
3. The trapdoor is the secret key of the PKE
4. In normal phase - use the leaked string as initial state of an underlying PRG

Immunization models

- Public immunization ✗
- Semi-private immunization
- Private immunization

Immunization models

- Public immunization ✗
- Semi-private immunization
- Private immunization

Private randomness

Observation

$f_{\text{seed}}(R) = PRF_{\text{seed}}(R)$ is secure immunization in private model.

Private randomness

Observation

$f_{\text{seed}}(R) = PRF_{\text{seed}}(R)$ is secure immunization in private model.

- Unsatisfactory.

Private randomness

Observation

$f_{\text{seed}}(R) = PRF_{\text{seed}}(R)$ is secure immunization in private model.

- Unsatisfactory.
- If users had access to a backdoor-less PRF, then instead of using it to immunize a backdoored PRG, they could use the PRF itself for pseudorandomness.

Private randomness

Observation

$f_{\text{seed}}(R) = PRF_{\text{seed}}(R)$ is secure immunization in private model.

- Unsatisfactory.
- If users had access to a backdoor-less PRF, then instead of using it to immunize a backdoored PRG, they could use the PRF itself for pseudorandomness.
- Goal - explore functions weaker than PRF.

Private randomness

Observation

$f_{\text{seed}}(R) = PRF_{\text{seed}}(R)$ is secure immunization in private model.

- Unsatisfactory.
- If users had access to a backdoor-less PRF, then instead of using it to immunize a backdoored PRG, they could use the PRF itself for pseudorandomness.
- Goal - explore functions weaker than PRF.
- See paper for initial negative results

Immunization models

- Public immunization ✗
- Semi-private immunization
- Private immunization ?

Immunization models

- Public immunization ✗
- Semi-private immunization
- Private immunization ?

Immunization models

- Public immunization ✗
- Semi-private immunization ✓
- Private immunization ?

Semi private randomness

- Recall G does not know seed of f_{seed}, but the attacker A does
- PRF does not work as seed is not secret
- Natural Immunization function:

$$f_{\text{seed}}(R) = RO(R||seed)$$

Theorem

$f_{\text{seed}}(R) = RO(R||seed)$ is secure immunization in the semi-private model

Positive result in ROM

Intuition:

- PRG outputs should have entropy even given the trapdoor

Positive result in ROM

Intuition:

- PRG outputs should have entropy even given the trapdoor
  - If outputs do not have entropy, there are collisions - can be publicly detected.

Positive result in ROM

Intuition:

- PRG outputs should have entropy even given the trapdoor
  - If outputs do not have entropy, there are collisions - can be publicly detected.
- Collision entropy $\implies$ min entropy

Positive result in ROM

Intuition:

- PRG outputs should have entropy even given the trapdoor
  - If outputs do not have entropy, there are collisions - can be publicly detected.
- Collision entropy $\implies$ min entropy
- RO extracts pseudorandomness from min entropy

Positive result in ROM

- Advantage in $\mathcal{G}_{\text{dist}}$ after immunization:

$$\text{Adv}(A_{sk}) \approx q_{RO} q_{PRG} \sqrt{\text{Adv}(D_{pk})}$$

- Open question - Is this poor dependence inherent?
- In the standard model - replacing RO with a UCE (Bellare et al 2013) secure hash function is a secure immunization.
  - Strong standard model assumption, but does not come under the impossibility results (Brzuska, Farshim and Mittelbach 2014)

Summary and Further questions

- Definitional framework of Backdoored PRGs.
- Equivalence of backdoored PRGs and public-key encryption schemes with pseudorandom ciphertexts
- Investigate countermeasures to BPRGs - immunizers
  - (In)effectiveness of countermeasures
  - Provably secure solution
- Open:
  - Immunization in Private model - is PRF necessary?
  - Semi-private - Positive result based on more standard assumptions?

Thank You

Thank you!