FPAKE: Fuzzy Password-Authenticated Key Exchange

Sophia Yakoubov
joint work with
Pierre-Alain Dupont, Julia Hesse,
David Pointcheval, Leonid Reyzin

Motivation

p@$$w0rd12

- Want: secure communication
- Over insecure, unauthenticated channel
- Shared secret: password
- The password is…
  - Low-entropy

p@$$w0rd12

Motivation

p@$$w0rd12

- Want: secure communication
- Over insecure, unauthenticated channel
- Shared secret: password
- The password is…
  - Low-entropy
  - Possibly noisy

p@$$w@rd12

Motivation

p@$$w0rd12

- Goal: Agree on high-entropy cryptographic key
- Man-in-the-middle security: Nothing leaks about…
  - Password
  - Key

p@$$w@rd12

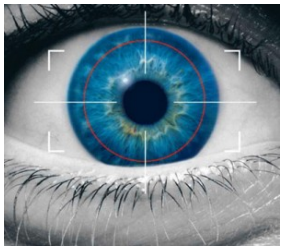
Applications

p@$$w0rd12

- Mistyped passwords
  e.g. [Chatterjee-Athalye-Akhawe-Juels-Ristenpart-16]

p@$$w@rd12

Applications: Not Just Passwords!



- Mistyped passwords
- Biometric authentication

Bob has a resource Alice is trying to access

Applications:
Not Just Passwords!

- Mistyped passwords
- Biometric authentication
- Location-based authentication
  e.g. [Han-Harishankar-Wang-Chung-Tague-17]

"radiation stinks
has 3 potholes"

"radiator springs
has 4 potholes"

Related Work

are the passwords **low-entropy**?
low-entropy: can hit correct password by brute-force enumeration

do the passwords have **noise**?

Related Work

are the passwords **low-entropy**?

do the passwords have **noise**?

	Low-entropy password	High-entropy password
Exact match		
Fuzzy match		

Related Work

	Low-entropy password	High-entropy password
Exact match		privacy amplification [Maurer-97, …]
Fuzzy match		

Related Work

	Low-entropy password (no leakage allowed)	High-entropy password (some leakage ok)
Exact match		privacy amplification [Maurer-97, …]
Fuzzy match		

Related Work

	Low-entropy password (no leakage allowed)	High-entropy password (some leakage ok)
Exact match	PAKE [Bellare-Pointcheval-Rogaway-00, Boyko-MacKenzie-Patel-00, …]	privacy amplification [Maurer-97, …]
Fuzzy match		

Secure against off-line dictionary attacks against the password

Related Work

	Low-entropy password (no leakage allowed)	High-entropy password (some leakage ok)
Exact match	PAKE [Bellare-Pointcheval-Rogaway-00, Boyko-MacKenzie-Patel-00, …]	privacy amplification [Maurer-97, …]
Fuzzy match		information reconciliation [Renner-Wolf-04, …] robust fuzzy extractors [Boyen-Dodis-Katz-Ostrovsky-Smith-05, …]

Related Work

	Low-entropy password (no leakage allowed)	High-entropy password (some leakage ok)
Exact match	PAKE [Bellare-Pointcheval-Rogaway-00, Boyko-MacKenzie-Patel-00, …]	privacy amplification [Maurer-97, …]
Fuzzy match	generic multi-party computation without authenticated channels [Barak-Canetti-Lindell-Pass-Rabin-05]	information reconciliation [Renner-Wolf-04, …] robust fuzzy extractors [Boyen-Dodis-Katz-Ostrovsky-Smith-05, …]

Fuzzy PAKE

	Low-entropy password (no leakage allowed)	High-entropy password (some leakage ok)
Exact match	PAKE [Bellare-Pointcheval-Rogaway-00, Boyko-MacKenzie-Patel-00, …]	privacy amplification [Maurer-97, …]
Fuzzy match	This paper - Fuzzy PAKE	information reconciliation [Renner-Wolf-04, …] robust fuzzy extractors [Boyen-Dodis-Katz-Ostrovsky-Smith-05, …]

Our Contributions

- Security definition
- Two efficient constructions

of Fuzzy Password Authenticated Key Exchange

Fuzzy PAKE

	Low-entropy password (no leakage allowed)	High-entropy password (some leakage ok)
Exact match	PAKE [Bellare-Pointcheval-Rogaway-00, Boyko-MacKenzie-Patel-00, …]	privacy amplification [Maurer-97, …]
Fuzzy match	This paper - Fuzzy PAKE	information reconciliation [Renner-Wolf-04, …] robust fuzzy extractors [Boyen-Dodis-Katz-Ostrovsky-Smith-05, …]

Our Contributions

- Security definition
- Two efficient constructions

of Fuzzy Password Authenticated Key Exchange

FPAKE Definition

I learned nothing about the key or the passwords

p@$$w0rd12
high-entropy key k

If the passwords are “similar enough”
the parties agree on a key

p@$$w@rd12
high-entropy key k

FPAKE Definition

I learned nothing about Bob's password

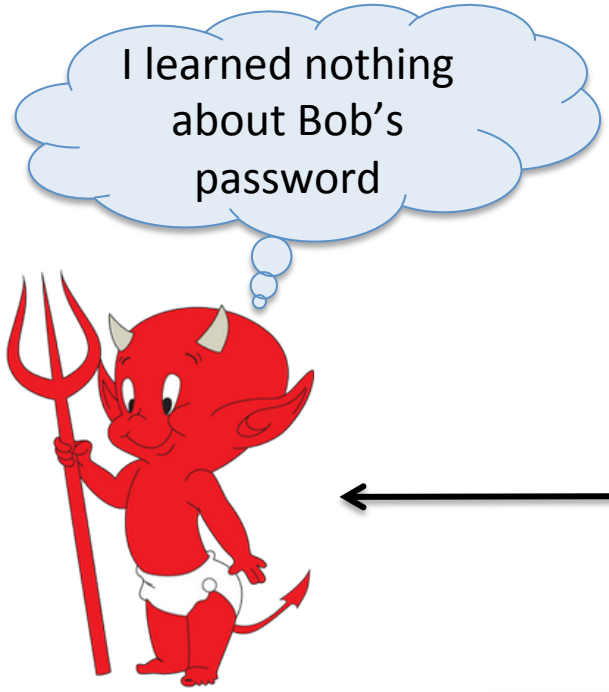
"curiouser"

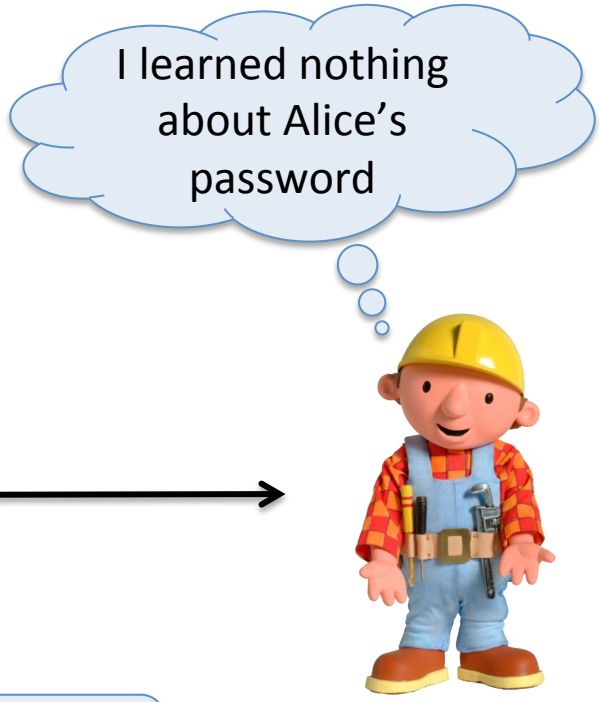
I learned nothing about Alice's password

"can we fix it?"

If the passwords are not "similar enough" the parties do not agree on a key

FPAKE Definition





"curiouser"

If the passwords are not "similar enough" the parties do not agree on a key

"can we fix it?"

- Definitional Goals:
  - Composability (essential for key agreement!)
  - Security against offline dictionary attacks by:
    - Malicious participant
    - Man in the Middle
- Approach: Generalize UC functionality for PAKE [Canetti-Halevi-Katz-Lindell-MacKenzie-05]

Fuzzy PAKE

	Low-entropy password (no leakage allowed)	High-entropy password (some leakage ok)
Exact match	PAKE [Bellare-Pointcheval-Rogaway-00, Boyko-MacKenzie-Patel-00, …]	privacy amplification [Maurer-97, …]
Fuzzy match	This paper - Fuzzy PAKE	information reconciliation [Renner-Wolf-04, …] robust fuzzy extractors [Boyen-Dodis-Katz-Ostrovsky-Smith-05, …]

Our Contributions

- UC security definition
- Two efficient constructions

of Fuzzy Password Authenticated Key Exchange

Constructions

- Using PAKE + Robust Secret Sharing (RSS)
- Using Yao's Garbled Circuits (YGC)
  - Not generic 2PC - specialized, secure variant of dual execution!

FPAKE construction	PAKE/RSS	Yao's Garbled Circuits
Notion of similarity	Hamming	Any
# rounds	2	5
# exponentiations	2n + constant	3n + constant

n = number of password characters

Constructions

- Using PAKE + Robust Secret Sharing (RSS)
- Using Yao's Garbled Circuits (YGC)
  - Not generic 2PC - specialized, secure variant of dual execution!

FPAKE construction	PAKE/RSS	Yao's Garbled Circuits
Notion of similarity	Hamming (same-length passwords)	Any
# rounds	2	5
# exponentiations	2n + constant	3n + constant

n = number of password characters

Fuzzy PAKE from PAKE/RSS

$\mathbf{pw_A}$= p@$$w0rd12

$\mathbf{pw_B}$= p@$$w@rd12

Pick a random session key **K**

"magical encryption" of **K** that tolerates errors in the encryption key $\mathbf{pw_A}$

C = Enc(key=$\mathbf{pw_A}$, msg=**K**)

K = Dec(key=$\mathbf{pw_B}$, **C**)

- **Problem**: ciphertext enables offline dictionary attack!
- **Solution**:
  1. **expand** each character using PAKE!
  2. **magical encryption** using resulting **character keys**

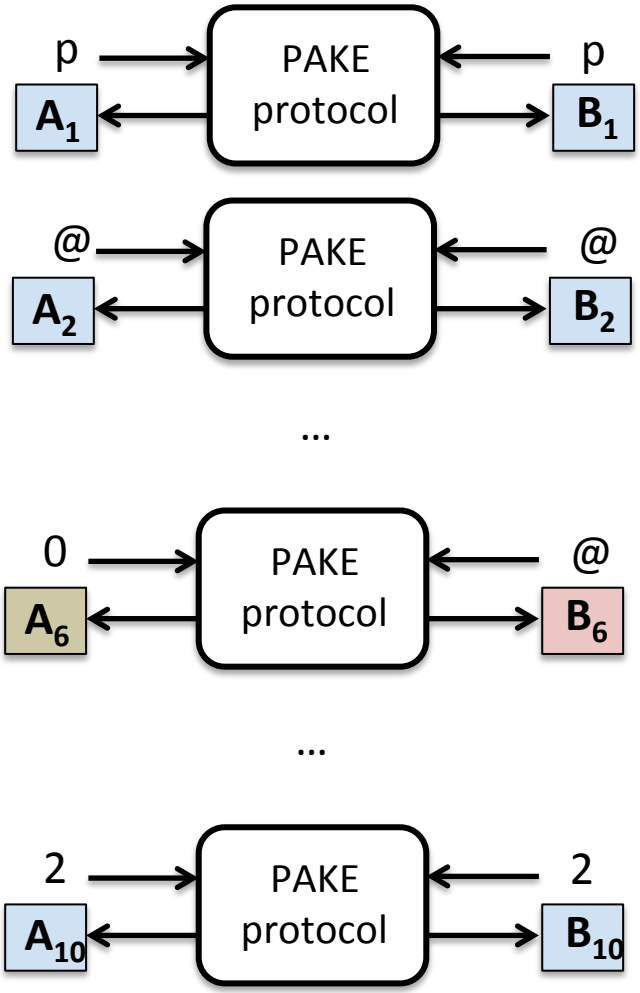
Fuzzy PAKE from PAKE/RSS: the Expansion Step

$\mathbf{pw_A}$= p@$$w0rd12

$\mathbf{A}$= | A_1 | A_2 | A_3 | A_4 | A_5 | A_6 | A_7 | A_8 | A_9 | A_{10} |

$\mathbf{pw_B}$= p@$$w@rd12

$\mathbf{B}$= | B_1 | B_2 | B_3 | B_4 | B_5 | B_6 | B_7 | B_8 | B_9 | B_{10} |



Fuzzy PAKE from PAKE/RSS: the Expansion Step

$\mathbf{pw_A}$= p@$$w0rd12

A= | A_1 | A_2 | A_3 | A_4 | A_5 | A_6 | A_7 | A_8 | A_9 | A_{10} |

$\mathbf{pw_B}$= p@$$w@rd12

B= | B_1 | B_2 | B_3 | B_4 | B_5 | B_6 | B_7 | B_8 | B_9 | B_{10} |

- It is important to hide whether agreement happened!
- Otherwise, locations of matching characters will leak
- **New UC definition**: implicit-only PAKE
  - no key confirmation: participants do not know whether agreement was successful
  - "EKE2" [Bellare-Pointcheval-Rogaway-00] is implicit-only

Fuzzy PAKE from PAKE/RSS

$\mathbf{pw_A}$= p@$$w0rd12

A= | A_1 | A_2 | A_3 | A_4 | A_5 | A_6 | A_7 | A_8 | A_9 | A_{10} |

Pick a random session key **K**

$\mathbf{pw_B}$= p@$$w@rd12

B= | B_1 | B_2 | B_3 | B_4 | B_5 | B_6 | B_7 | B_8 | B_9 | B_{10} |

"magical encryption" of **K** that tolerates errors in the encryption key $\mathbf{pw_A}$

C = Enc(key=$\mathbf{pw_A}$, msg=**K**)

K = Dec(key=$\mathbf{pw_B}$, **C**)

Fuzzy PAKE from PAKE/RSS

$\mathbf{pw_A}$= p@$$w0rd12

A= | A_1 | A_2 | A_3 | A_4 | A_5 | A_6 | A_7 | A_8 | A_9 | A_{10} |

Pick a random session key **K**

$\mathbf{pw_B}$= p@$$w@rd12

B= | B_1 | B_2 | B_3 | B_4 | B_5 | B_6 | B_7 | B_8 | B_9 | B_{10} |

"magical encryption" of **K** that tolerates errors in the encryption key **A**

C = Enc(key=**A**, msg=**K**)

K = Dec(key=**B**, **C**)

Fuzzy PAKE from PAKE/RSS

$\mathbf{pw_A}$= p@$$w0rd12

A= | A_1 | A_2 | A_3 | A_4 | A_5 | A_6 | A_7 | A_8 | A_9 | A_{10} |

Pick a random session key **K**

$\mathbf{pw_B}$= p@$$w@rd12

B= | B_1 | B_2 | B_3 | B_4 | B_5 | B_6 | B_7 | B_8 | B_9 | B_{10} |

Robust Secret Sharing of **K** + One Time Pad

C = Enc(key=**A**, msg=**K**)

Similar to Code-Offset [Juels-Watenberg-02]

K = Dec(key=**B**, **C**)

Constructions

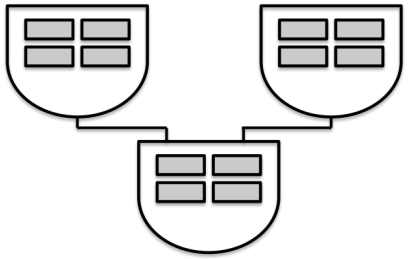
- Using PAKE + Robust Secret Sharing (RSS)
- Using Yao's Garbled Circuits (YGC)
  - Not generic 2PC - specialized, secure variant of dual execution!

FPAKE construction	PAKE/Secret Sharing	Yao's Garbled Circuits
Notion of similarity	Hamming	Any
# rounds	2	5
# exponentiations	2n + constant	3n + constant

n = number of password characters

Yao's Garbled Circuits for FPAKE

garbler



evaluator

output

Yao's Garbled Circuits for FPAKE

pw_G = p@$$w0rd12

pw_E = p@$$w@rd12

Circuit determining whether pw_G and pw_E are "close enough"; outputs session key if yes

garbler

evaluator

output

Yao's Garbled Circuits for FPAKE

pw_G = p@$$w0rd12

pw_E = p@$$w@rd12

Circuit determining whether pw_G and pw_E are "close enough"; outputs session key if yes

output

semi-honest garbler

malicious evaluator

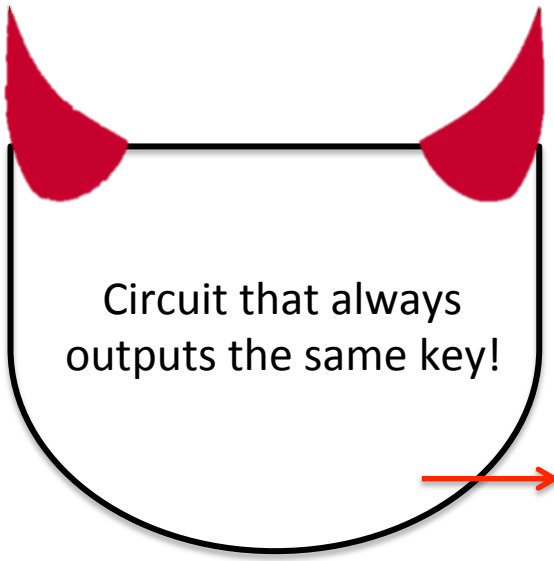
Yao's Garbled Circuits are an asymmetric 2PC protocol: they are secure against a malicious evaluator, but only against a semi-honest garbler

Yao's Garbled Circuits for FPAKE

pw_G = p@$$w0rd12

~~semi-honest~~ garbler

pw_E = p@$$w@rd12



malicious evaluator

Yao's Garbled Circuits are an asymmetric 2PC protocol: they are secure against a malicious evaluator, but only against a semi-honest garbler

From Semi-Honest to Malicious

Correctness	Privacy	Computation Overhead

From Semi-Honest to Malicious

Transformation	Correctness	Privacy	Computation Overhead
None			
Commit-and-Prove			
Cut-and-Choose			
Gate-wise Cut-and-Choose			(including pre-processing)
…			

From Semi-Honest to Malicious

Transformation	Correctness	Privacy	Computation Overhead
None			
Commit-and-Prove			
Cut-and-Choose			
Gate-wise Cut-and-Choose			(including pre-processing)
…			
Dual Execution [Mohassel-Franklin-06, Huang-Katz-Evans-12]		1 bit leakage	Only 2x!

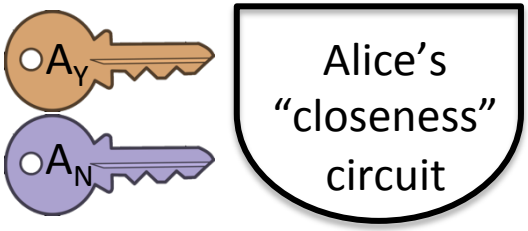
1 bit of leakage about a low-entropy password can be crucial!

We modify dual execution specifically for Fuzzy PAKE to avoid leakage when it matters

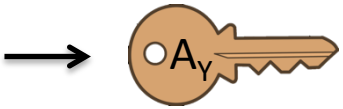
pw_G = p@$$w0rd12

pw_E = p@$$w@rd12

Dual Execution for FPAKE



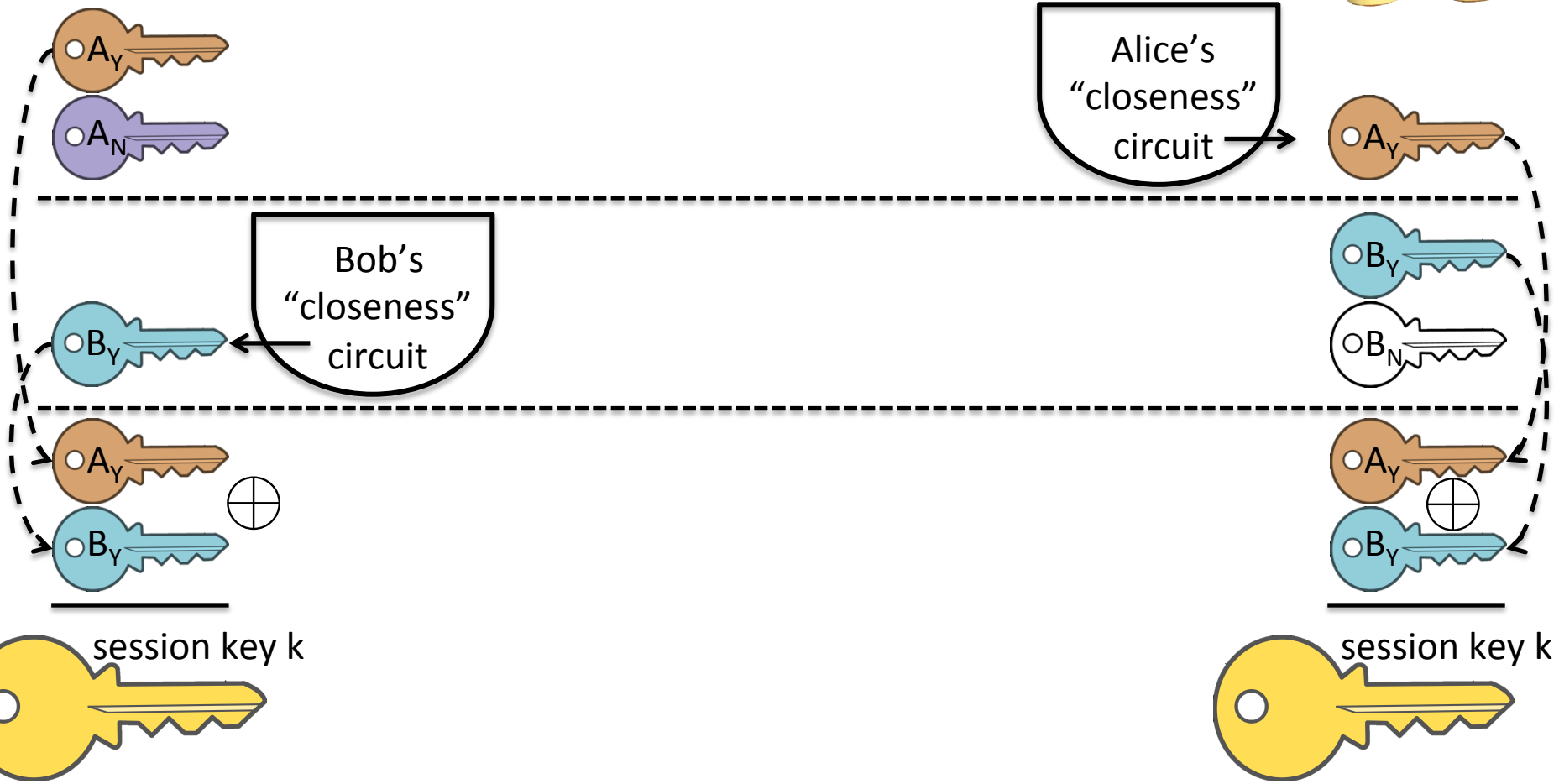
circuit that outputs yes/no yes/no key



pw_G = p@$$w0rd12

pw_E = p@$$w@rd12

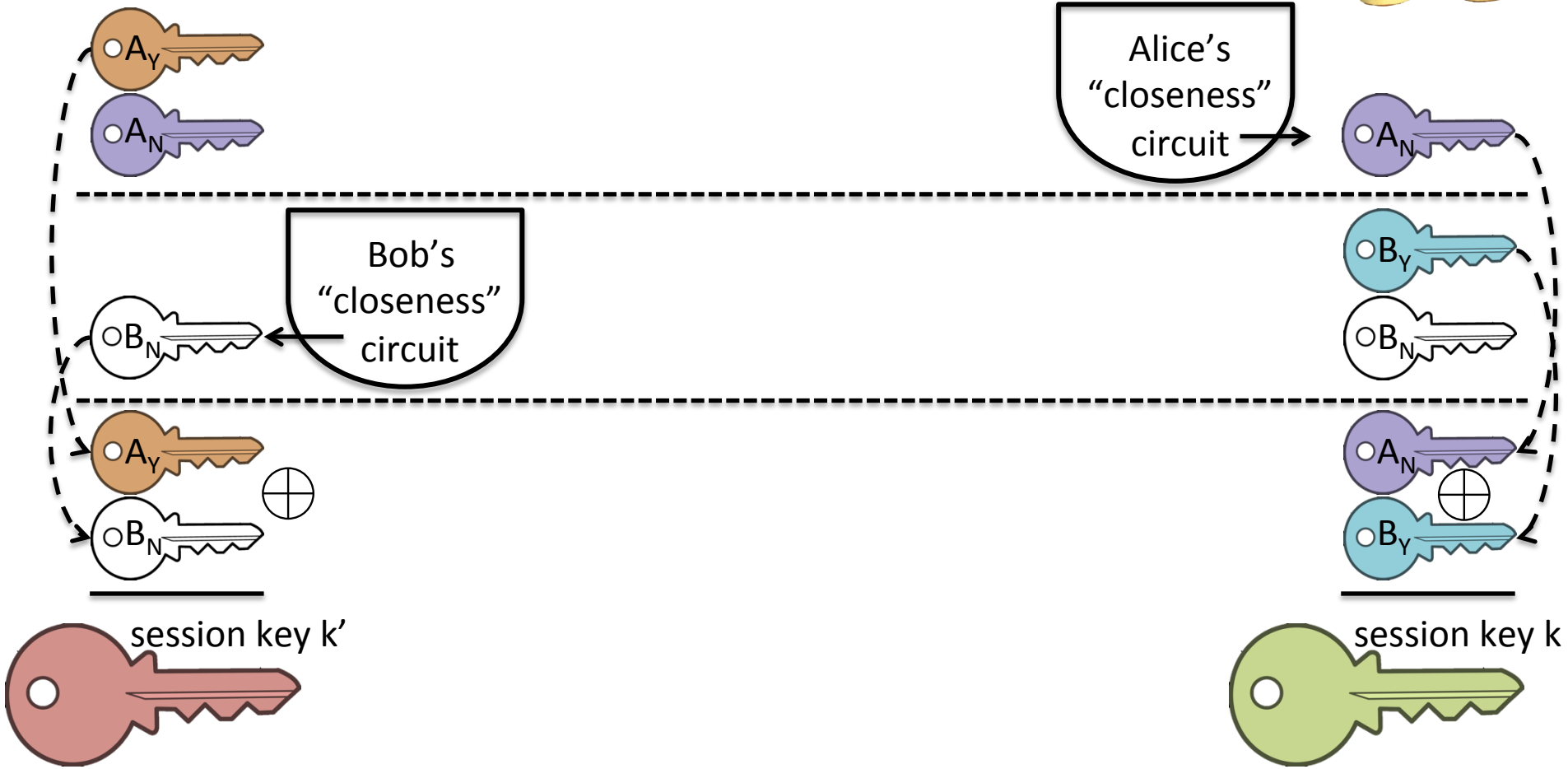
Dual Execution for FPAKE



pw_G = “curiouser”

pw_E = “can we fix it”

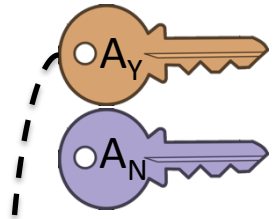
Dual Execution for FPAKE

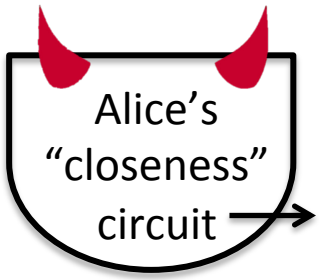


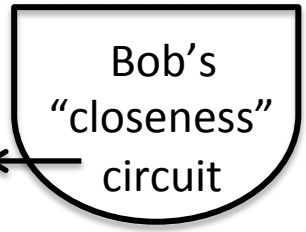
Dual Execution for FPAKE

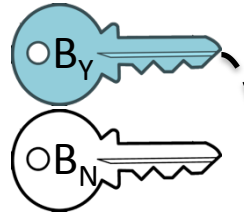
pw_G = “curiouser”

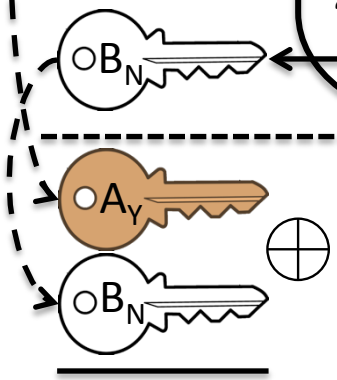
pw_E = “can we fix it”

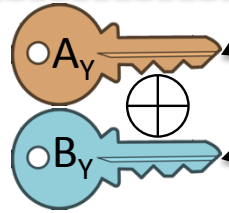


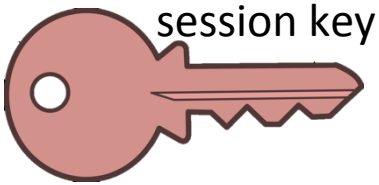












session key k’

session key k

Dual Execution for FPAKE

pw_G = “curiouser”

pw_E = “can we fix it”

No clue about B_N !

A_Y A_N

Alice’s “closeness” circuit

A_Y

B_Y B_N

Bob’s “closeness” circuit

B_N

A_Y B_N

A_Y B_Y

session key k’

session key k

Dual Execution: Privacy-Correctness Tradeoff for Boolean Functions

[MF'06, HKE'12] Dual Execution	Correct output	Comp. output	Privacy
	"yes"	"yes" or "cheating"	1-bit leakage
	"no"	"no" or "cheating"	1-bit leakage

Our FPAKE Dual Execution	Correct output	Comp. output	Privacy
	"yes"	"yes" or "no"	1-bit leakage
	"no"	"no"	yes

This is the perfect tradeoff for fuzzy PAKE!

- Only care about security against adversary who doesn't know a close-enough password – the "no" case

Conclusion

	Low-entropy password	High-entropy password
Exact match	PAKE	privacy amplification
Fuzzy match	This paper - Fuzzy PAKE	information reconciliation, robust fuzzy extractors

Our Contributions

- UC security definition of Fuzzy PAKE
- Two efficient constructions:
  - PAKE + Robust Secret Sharing (more efficient)
  - Yao's Garbled Circuits (more general)