Searching on Encrypted Data

Iwan Richards and Harrison Nicholls

The Problem:

Want to be able to search for things in data without revealing anything about the data, or about our searches. We can give a third party, S (the server) the responsibility of facilitating this search, and model S as the adversary who wants to learn about our searches. The searches correspond with specific keywords.

Overview

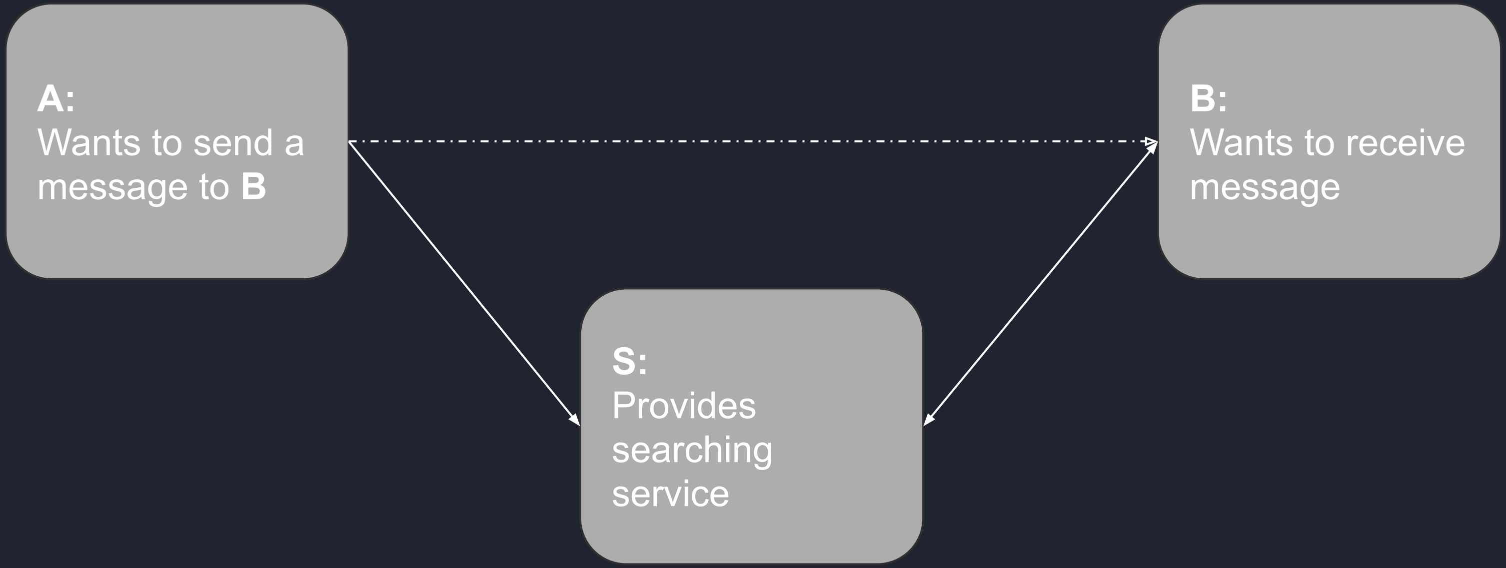
PEKS

Boneh et. al.
Baek et al.
Kim et al.

SBS-PED

Proposed: KwdMAC

The Problem: Parties in PEKS



Syntax

$$PEKS = (Kg, PEKS, Td, Test)$$

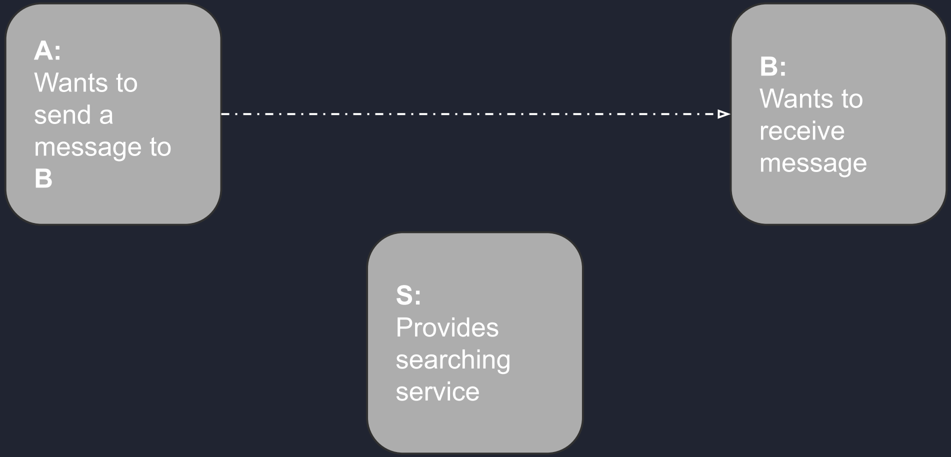
$$(k_p, k_s) \xleftarrow{\$} \mathcal{KG}(1^k) | k \in N$$

$$C \xleftarrow{\$} \mathcal{PEKS}^H(k_p, w)$$

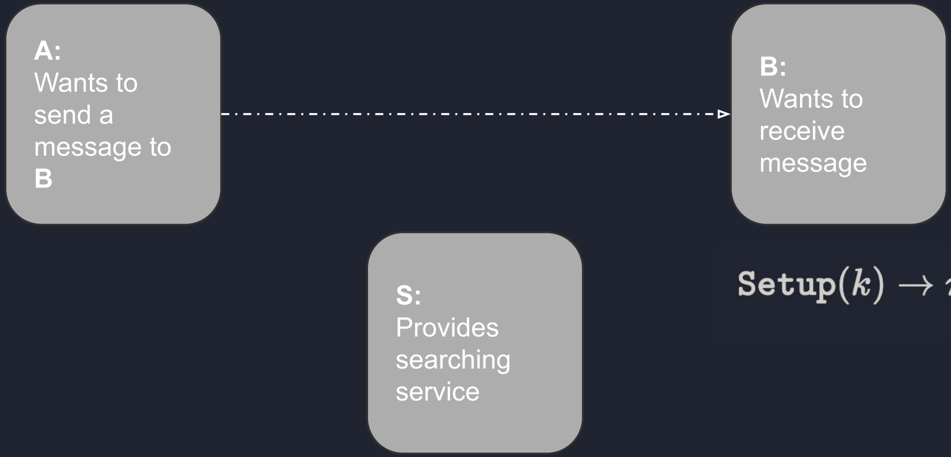
$$t_w \xleftarrow{\$} \mathcal{T}d^H(k_s, w)$$

$$b \xleftarrow{\$} \mathcal{T}est^H(t_w, C)$$

Functions


A:
Wants to send a message to **B**
B:
Wants to receive message
S:
Provides searching service

Functions



$$\mathtt{Setup}(k) \rightarrow \pi,\ \mathtt{KeyGen}(\pi) \rightarrow (A_{pub}, A_{priv})$$

Functions

A:
Wants to send a message to **B**

c

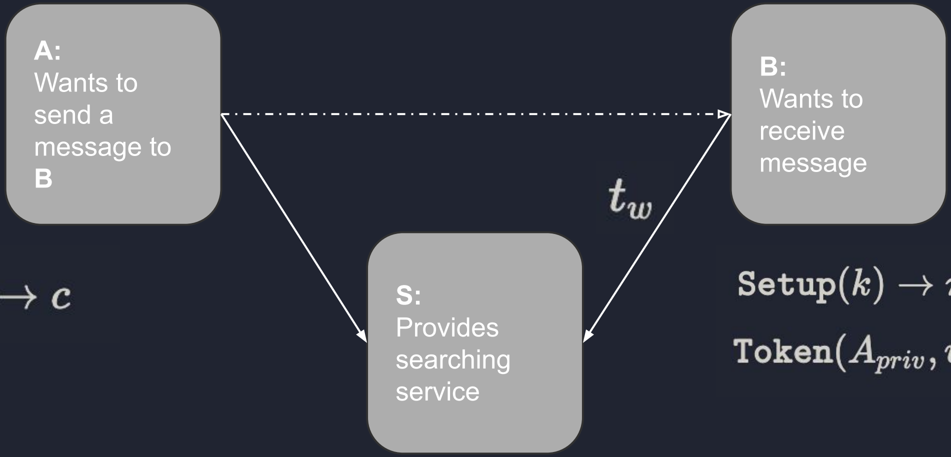
S:
Provides searching service

B:
Wants to receive message

$$\texttt{PEKS}(A_{pub}, w, \pi) \rightarrow c$$

$$\texttt{Setup}(k) \rightarrow \pi, \texttt{KeyGen}(\pi) \rightarrow (A_{pub}, A_{priv})$$

Functions

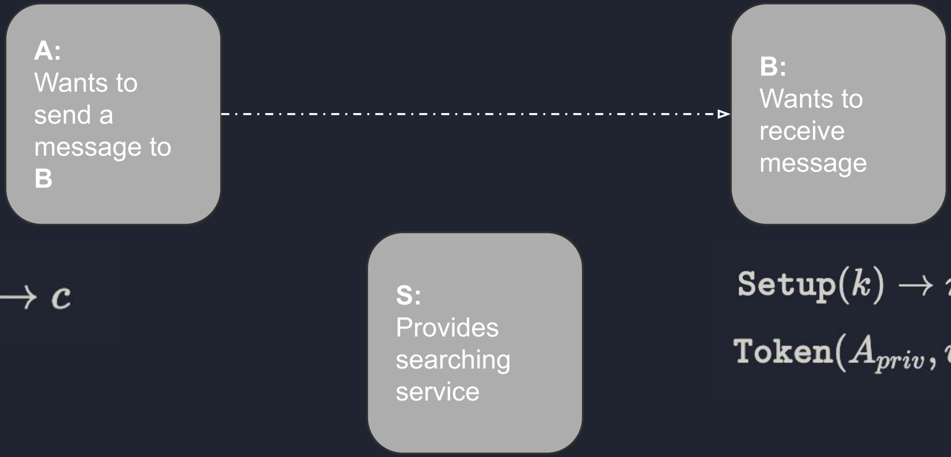


$$\texttt{PEKS}(A_{pub}, w, \pi) \rightarrow c$$

$$\texttt{Setup}(k) \rightarrow \pi, \texttt{KeyGen}(\pi) \rightarrow (A_{pub}, A_{priv})$$

$$\texttt{Token}(A_{priv}, w, \pi) \rightarrow t_w$$

Functions



$\mathtt{PEKS}(A_{pub}, w, \pi) \rightarrow c$

$\mathtt{Setup}(k) \rightarrow \pi$, $\mathtt{KeyGen}(\pi) \rightarrow (A_{pub}, A_{priv})$

$\mathtt{Token}(A_{priv}, w, \pi) \rightarrow t_w$

$Test(Apub, C, t_w, \pi) \rightarrow \mathtt{yes/no}$

Functions

A:
Wants to send a message to **B**

$$\texttt{PEKS}(A_{pub}, w, \pi) \rightarrow c$$

B:
Wants to receive message

$$\texttt{Setup}(k) \rightarrow \pi,\ \texttt{KeyGen}(\pi) \rightarrow (A_{pub}, A_{priv})$$

$$\texttt{Token}(A_{priv}, w, \pi) \rightarrow t_w$$

c

S:
Provides searching service

$$Test(Apub, C, t_w, \pi) \rightarrow \texttt{yes}$$

Generic Dictionary attack on PEKS

A generic dictionary attack on PEKS may be performed by any party with access to t_w including the search server. Additionally the attacker has the extra advantage of using context to guess tags, such as guessing business names if targeting stock brokers. Effectively the scheme performs:

$$Test(A_{pub}, C, t_w, \pi) = \text{``}yes\text{''}, \text{with } C' = PEKS(A_{pub}, w, \pi)$$

And an attacker performs:

$$Test(A_{pub}, C', t_w, \pi) = \text{``}yes\text{''}, \text{with } C' = PEKS(A_{pub}, w', \pi)$$

Proposed solution

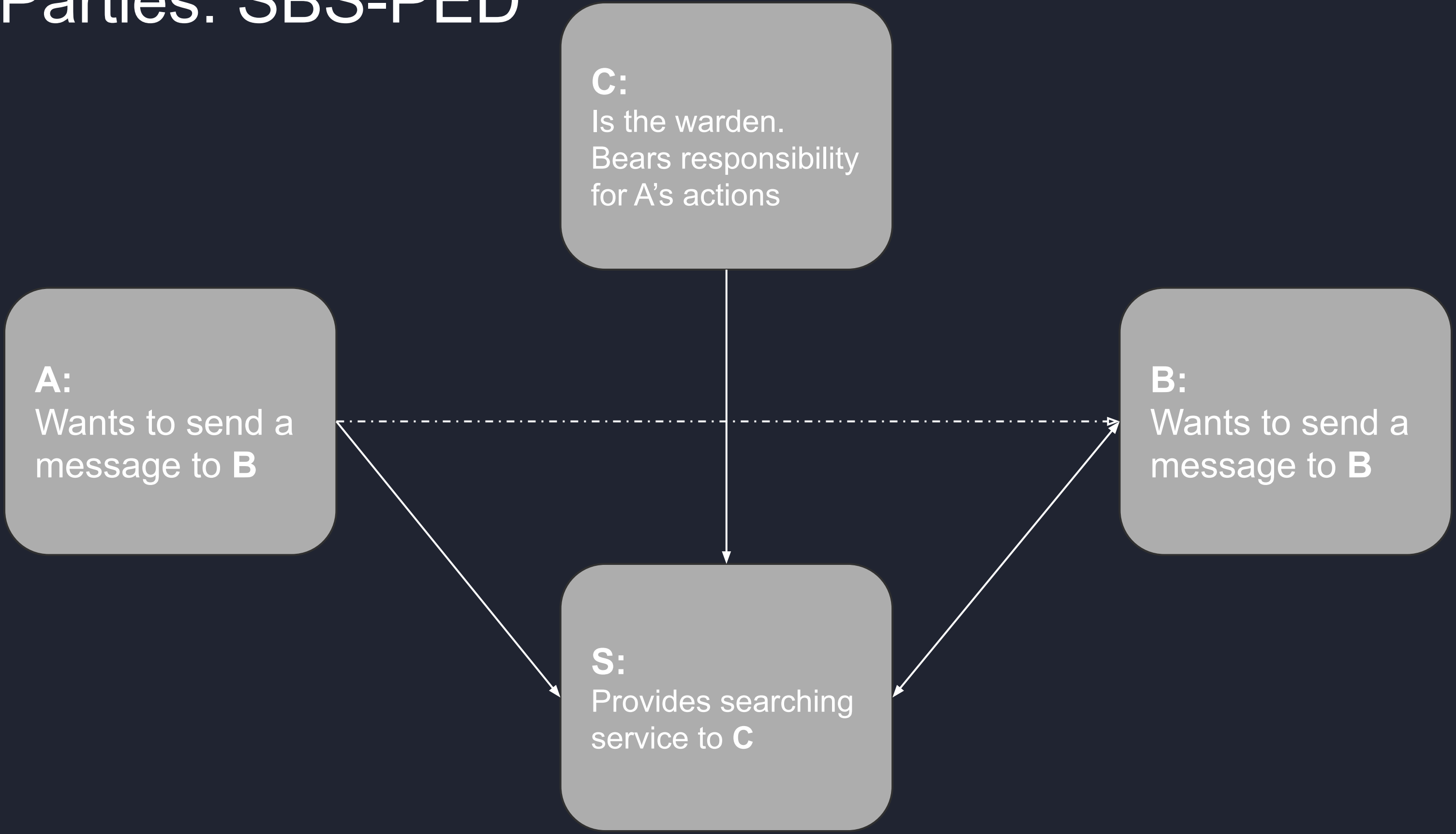
PEKS

Boneh et. al.
Baek et al.
Kim et al.

SBS-PED

Proposed: KwdMAC

The Parties: SBS-PED



Functions

$$\texttt{Setup}(\pi_s, \pi_f) \rightarrow \pi$$

$$\texttt{User-Reg}(A, P_{auth}, \pi) \rightarrow r_A = (\pi, k_A)$$

C:
Is the warden.
Bears responsibility for A’s actions

A:
Wants to send a message to **B**

B:
Wants to send a message to **B**

S:
Provides searching service to **C**

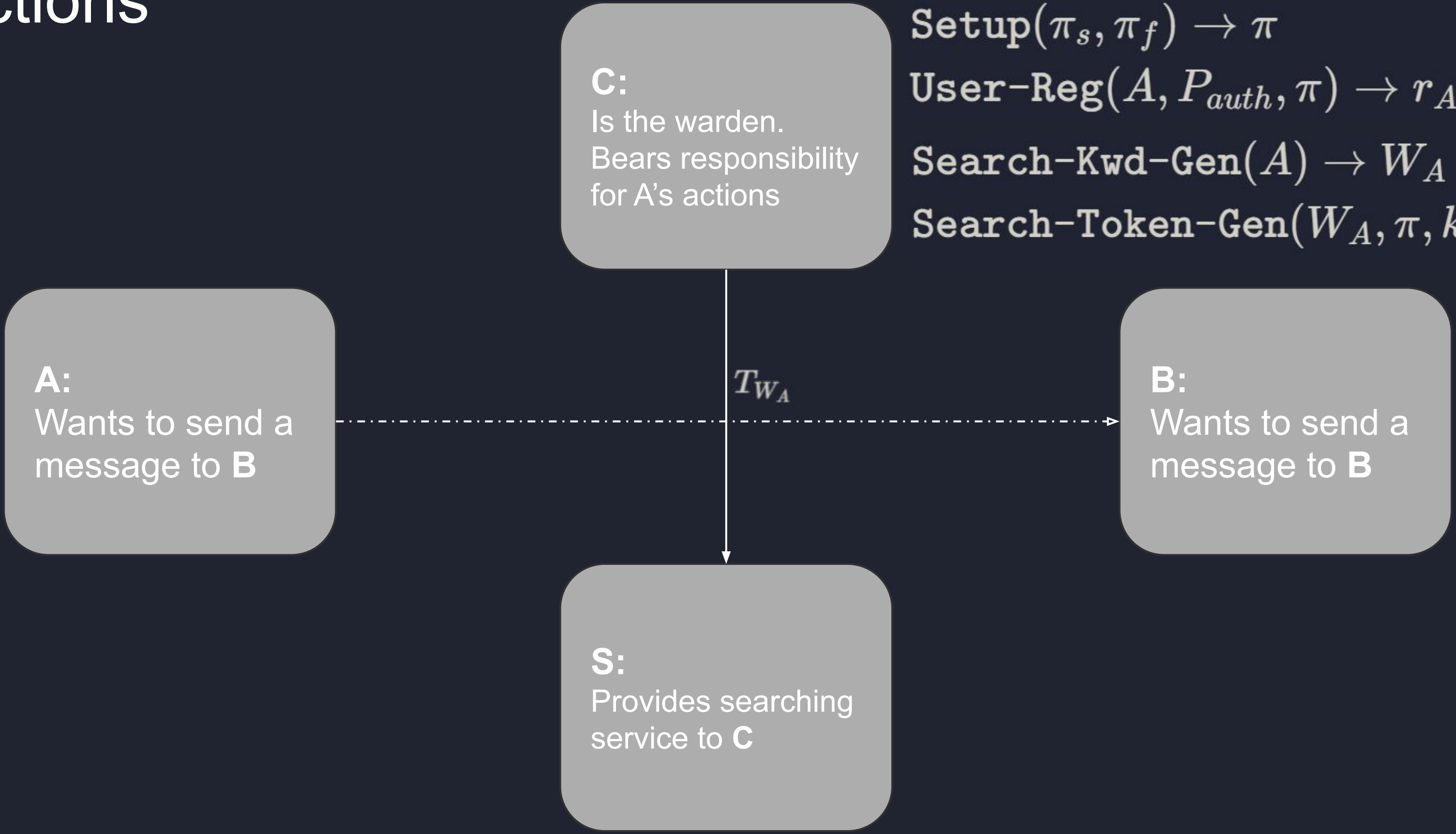
Functions

$$\texttt{Setup}(\pi_s, \pi_f) \to \pi$$

$$\texttt{User-Reg}(A, P_{auth}, \pi) \to r_A = (\pi, k_A)$$

$$\texttt{Search-Kwd-Gen}(A) \to W_A$$

$$\texttt{Search-Token-Gen}(W_A, \pi, k_A) \to T_{W_A}$$



Functions

C:
Is the warden.
Bears responsibility for A's actions

$$\mathtt{Setup}(\pi_s, \pi_f) \rightarrow \pi$$

$$\mathtt{User\text{-}Reg}(A, P_{auth}, \pi) \rightarrow r_A = (\pi, k_A)$$

$$\mathtt{Search\text{-}Kwd\text{-}Gen}(A) \rightarrow W_A$$

$$\mathtt{Search\text{-}Token\text{-}Gen}(W_A, \pi, k_A) \rightarrow T_{W_A}$$

A:
Wants to send a message to **B**

B:
Wants to send a message to **B**

c

$$\mathtt{Encrypt}(m, Pub_B, \pi, k_A) \rightarrow c$$

S:
Provides searching service to **C**

Functions

C:
Is the warden.
Bears responsibility for A's actions

$\texttt{Setup}(\pi_s, \pi_f) \rightarrow \pi$

$\texttt{User-Reg}(A, P_{auth}, \pi) \rightarrow r_A = (\pi, k_A)$

$\texttt{Search-Kwd-Gen}(A) \rightarrow W_A$

$\texttt{Search-Token-Gen}(W_A, \pi, k_A) \rightarrow T_{W_A}$

A:
Wants to send a message to **B**

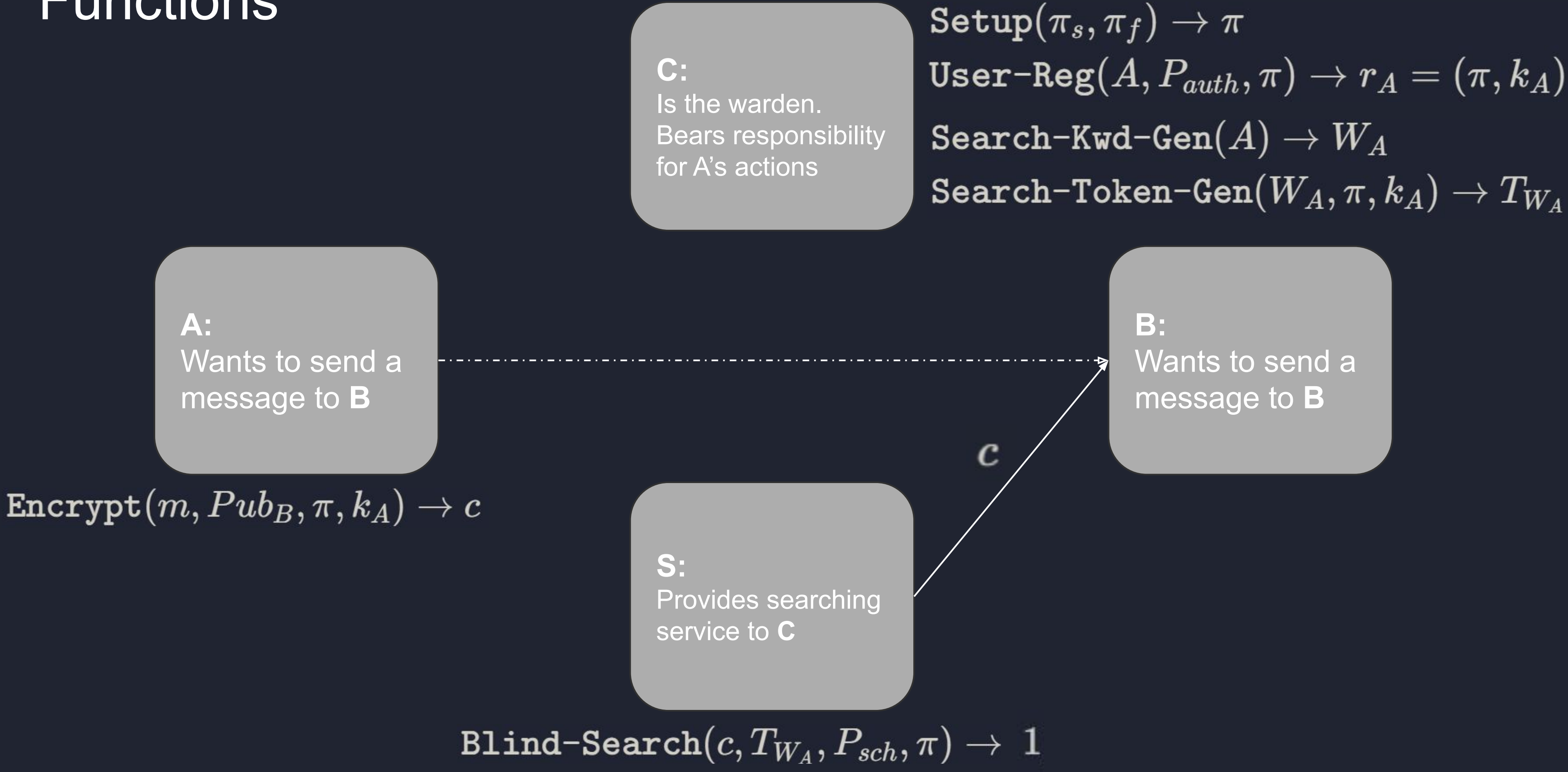
B:
Wants to send a message to **B**

$\texttt{Encrypt}(m, Pub_B, \pi, k_A) \rightarrow c$

S:
Provides searching service to **C**

$\texttt{Blind-Search}(c, T_{W_A}, P_{sch}, \pi) \rightarrow \{0,1\}$

Functions


C:
Is the warden.
Bears responsibility for A's actions
$\texttt{Setup}(\pi_s, \pi_f) \to \pi$
$\texttt{User-Reg}(A, P_{auth}, \pi) \to r_A = (\pi, k_A)$
$\texttt{Search-Kwd-Gen}(A) \to W_A$
$\texttt{Search-Token-Gen}(W_A, \pi, k_A) \to T_{W_A}$
A:
Wants to send a message to **B**
$\texttt{Encrypt}(m, Pub_B, \pi, k_A) \to c$
B:
Wants to send a message to **B**
c
S:
Provides searching service to **C**
$\texttt{Blind-Search}(c, T_{W_A}, P_{sch}, \pi) \to 1$

Kwd-MAC

$\texttt{Setup}(\pi_s, \pi_f) \to \pi$

$\texttt{User-Reg}(A, P_{auth}, \pi) \to r_A = (\pi, k_A)$

C runs $\texttt{Setup}$ to select a $\texttt{MAC}$ function $\texttt{H} : \{0,1\}^* \times \{0,1\}^{n_2} \to \{0,1\}^{n_3}$ where n_2 and n_3 are sufficiently large integers, also determined in $\texttt{Setup}$

$$\pi = (n_1, n_4, d_H, ID_S, Pub_S, Pub_C).$$

$\texttt{User-Reg}$: C authenticates A using policy P_{auth}

Kwd-MAC

$\texttt{Search-Kwd-Gen}(A) \to W_A$
$\texttt{Search-Token-Gen}(W_A, \pi, k_A) \to T_{W_A}$

Search-Kwd-Gen(A): is the process of C generating a dictionary W_A of search keywords, possibly specific to user A.

Search-Token-Gen(A): is the process of C building the data structure T_{W_A} encoding the set of search keywords $t_w = t_{(k_A,w)} = H_{k_A}(w)$ for each $w \in W_A$. C then gives T_{W_A} to S secretly.

Kwd-MAC

$\texttt{Encrypt}(m, Pub_B, \pi, k_A) \rightarrow c$

A computes $t_{(kA,v)} = H_{k_A}(v)$ for each $v \in V$
then:
$c = ([m]^B, (t_{(k_A,v)})v \in V, ID_B)$

Where $[m]^B$ denotes the asymmetric encryption of m with B's public key, and ID_B is a global identifier for B.

Kwd-MAC

$$\texttt{Blind-Search}(c, T_{W_A}, P_{sch}, \pi) \rightarrow \{0, 1\}$$

$\texttt{Blind-Search}(c, T_{W_A}, P_{sch}, \pi)$: is the process run by S to check if the chosen tokens T_{W_A} are present in c and a response based on the policy P_{sch} if tokens are present or absent.

A Potential Vulnerability in `SBS-PED`

A malicious party A' (For example, an employee of C with malicious intent) could disobey the policy specified by **`Encrypt`**:

Where normally:
$c = ([m]^B, (t_{(k_A,v)})v \in V, ID_B)$

A' could instead produce a message m that *does* contain violating keywords, but not generate $(t_{(k_A,v)})v \in V$ faithfully. It could instead include nothing and simply compute $c = ([m]^B, \varepsilon, ID_B)$, or $c = ([m]^B, (H_{k_A}(s))s \in R, ID_B)$ where R is a set of strings s chosen randomly.

This attack appears possible based on the construction of **`SBS-PED`**. The ideal direction for a solution to this appears to be modifying **`Blind-Search`** to check whether $(t_{(k_A,v)})v \in V$ is actually a function of the keywords in m, but it's very unclear how to do this given that S receives $[m]^B$, and cannot see m by the design of the scheme.