A Model for a Privacy-Preserving Event-Ambivalent Notification Scheme

Our Model

Here we present a first draft of our model for a privacy-preserving event-ambivalent notification scheme [1].

Our model requires that the event server is not able to record the sender of incoming messages. This can be achieved in practice either by assumption or requiring the client to do some sort of packet spoofing or use a communication protocol like Tor to blind the source.

Informally, an event-ambivalent notification scheme is a protocol between a client, an event server, and an alert server. The protocol consists of the following steps:

1. The client uses the `Setup` algorithm to generate a secret key `sk`.
2. Upon an event, the client uses `Identify` to generate an `id` given `sk` and the event. The client sends `id` and the event to the event server.
3. When the event server wants to send out an alert based on a specific event, it runs `Process` on the event `id` and sends the resulting `alert` to the alert server.
4. Upon receiving an `alert` the alert server runs `Label` to gain an address `addr` that it then places the alert in the mailbox labeled `addr`.
5. The client runs `Address` using `sk` to check receive a set of mailboxes `addrs` that they can periodically check for messages.

Provided the two servers do not collude, a privacy-preserving event-ambivalent notification scheme should ensure that:

1. The event server shouldn't be able to link different events stored by the same client together.
2. The alert server shouldn't learn anything from an identifier regarding the event it is tied to or what client generated it.
3. The client shouldn't be able to link a message in a mailbox back to a specific event.
4. The servers and any other clients shouldn't be able to link any mailboxes to a specific user.
5. Only a client who submitted an event can then check the resulting mailbox.

A rough draft of the formal definition of such a scheme:

Definition 1 A privacy-preserving event-ambivalent notification scheme is a tuple $\Pi = (\texttt{Setup}, \texttt{Identify}, \texttt{Process}, \texttt{Label}, \texttt{Address})$ of efficient algorithms:

[1] Design and presentation of this model inspired by Section 3.1 of [1].

- $\texttt{Setup}(1^\lambda) \rightarrow \texttt{sk}$*:*
- $\texttt{Identify}(\texttt{sk}, \texttt{event}) \rightarrow \texttt{id}$*:*
- $\texttt{Process}(\texttt{id}) \rightarrow \texttt{alert}$*:*
- $\texttt{Label}(\texttt{alert}) \rightarrow \texttt{addr}$*:*
- $\texttt{Address}(\texttt{sk}) \rightarrow \texttt{addrs}$*:*

Furthermore, Π must satisfy the following properties:
Correctness.

$$\Pr\left[\texttt{Label}(\texttt{alert}) \in \texttt{addr} : \begin{array}{r} \texttt{sk} \leftarrow \texttt{Setup}\left(1^\lambda\right) \\ \texttt{id} \leftarrow \texttt{Identify}\left(\texttt{sk}, \texttt{event}\right) \\ \texttt{alert} \leftarrow \texttt{Process}\left(\texttt{id}\right) \\ \texttt{addr} \leftarrow \texttt{Address}\left(\texttt{sk}\right) \end{array}\right] = 1$$

Security.

References

[1] H. Corrigan-Gibbs and D. Kogan. Private information retrieval with sublinear online time. In *EUROCRYPT*, May 2020.