Entangled Fuzzy Tags and their Applications

Sarah Jamie Lewis

February 2, 2021

1 Introduction

Fuzzy Message Detection[1] presents a mechanism for constructing probabilistic, cryptographic tags for use in metadata resistant systems.

Crucially, unlike bucketing based mechanisms, these tags allow parties to set their own false positive rates, creating a number of distinct classes of parties (from those who match and download everything, to those with a very low false positive rate who match and download only messages intended for them).

In this note we address several of the questions in the original paper, and outline a number of extensions to "fuzzytags" that allow the `FMD2` scheme to be used for multiparty broadcast and deniable sending through the construction of "entangled" tags, that are valid tags for more than one public key.

2 Choosing a false positive rate p

When different parties have different false positive rates the server can calculate the skew, Δ, between a party's ideal false positive rate p and observed false positive rate ϕ.

That skew leaks information, especially given certain message distributions. Specifically it leaks parties who receive a larger proportion of system messages than their ideal false positive rate i.e. for low false positive rates and high message volume for a specific receiver, the adversarial server can calculate a skew that leaks the recipient of individual messages - breaking privacy for that receiver.

It also removes those messages from the pool of messages that an adversarial server needs to consider for other receivers. Effectively reducing the anonymity set for everyone else.

2.1 Intersection Attacks

Without a significant number of parties downloading everything, any kind of differential attack breaks the fuzzy tag scheme, even for a small number of messages i.e. if you learn (through any means) that a specific set of messages are all likely for 1 party, you can compare them against the entire set of detection keys and very quickly isolate the intended recipient through intersection of sets of the keys that validate - in simulations[1] of 100-1000 parties,$\gamma = 24$, and $p = [1..4]$ it can take as little as 3 messages to isolate a detection key - even with all parties selecting fairly high false positive rates.

The corollary of the above being that in differential attacks your anonymity set is basically the number of users who download all messages. This has the interesting side effect: the more parties who download everything, the more the system can safely tolerate parties with small false-positive rates.

2.2 Parties that download everything

As such, parties who expect a large number of messages should choose to receive all messages for 2 reasons:

1. Even high false positive rates for power users result in information leaks to the server (due to the large Δ) i.e. a server can trivially learn what users are "power" users.

2. By choosing to receive all messages, power users don't sacrifice much in terms of bandwidth, but will provide cover traffic for parties

[1] A playground simulator for fuzzytags based can be found at `https://git.openprivacy.ca/openprivacy/fuzzytags-sim`

Draft

who receive a small number of messages and who want a lower false-positive rate.

3 Entangled Fuzzy Tags

Due to their unique properties, it is possible to forge tags that validate against multiple distinct detection keys.

The probability of generating a tag that validates against n distinct detection keys of max length l is $(2^{-l})^{n-1}$, i.e. a sender should expect to find a suitable tag in $\frac{1}{(2^{-l})^{n-1}}$ generations (see Figure:1 for a method to find suitable tags given a set of public keys.).

Because each party determines their own false positive rate, it may not always be necessary for a sender to generate a tag that will be guaranteed to match up to the system parameter γ, instead they could generate a partially entangled fuzzy tag that would validate against *any* detection key from one party, and only detection keys with high false positive tolerances for other parties.

3.1 Applications

In this section we outline a number of applications enabled by entangled fuzzy tags.

3.2 Multiparty Broadcast

Alice wants to send a message to Bob and Carol. She constructs a single tag that will validate against detection keys generated by both of them.

When an adversarial server matches the tag against all the keys it knows about it will discover that the tag matches both Bob and Carol (in addition to some number of false positives depending on the false positive rates of all the other parties using the server).

To construct such a tag Alice runs $\texttt{FlagEntangled}(\{\mathsf{pk}_{\text{bob}}, \mathsf{pk}_{\text{carol}}\})$.

The adversarial server will match the tag to the detection keys of both Bob and Carol. The server has no way of determining if the match is a broadcast to both parties, a unique message to one of Bob or Carol or a false positive for both.

$\texttt{FlagEntangled}(\kappa = \{\mathsf{pk}_1 \dots \mathsf{pk}_n\})$

1 : $g, h_{i1} \dots h_{i\gamma} \leftarrow \mathsf{pk}_i : \forall \mathsf{pk}_i \in \kappa$
2 : $r \leftarrow_\$ \mathbb{Z}_q$
3 : $u \leftarrow g^u$
4 : $z \leftarrow_\$ \mathbb{Z}_q$
5 : $w \leftarrow g^z$
6 : **for** $h \in \kappa$:
7 : **for** $j \in [\gamma]$:
8 : $k_{ij} = H(u \| h_j^r \| w)$
9 : **endfor**
10 : **endfor**
11 : **if** $k_0 = k_1 = k_2 \dots = k_n$
12 : // derive the tag as in the original protocol.
13 : // using k_{0j} as the key for part j.
14 : **else**
15 : // goto 4
16 : **endif**

Figure 1: Pseudocode for deriving FMD2-compatible tags that can be verified by multiple detection keys, each related to a distinct public key. All functions as defined in [1]. Several performance improvements are possible e.g. caching the result of h_j^r to avoid duplicate group operations, and testing key equality earlier.

3.3 Deniable Sending

Alice wants to send a message to Carol, but is concerned that Carol may have a detection key with too low false positive rate. Alice knows of a set of parties (and their public keys) who also use the adversarial server to send privacy messages. Alice searches for a tag that will validate against detection keys generated not only by Carol but a randomly selected party e.g. Eve.

When an adversarial server matches the tag against all the keys it knows about it will discover that the tag matches both Carol and Eve (in addition to some number of false positives depending on the false positive rates of all the other parties using the server).

Even if the server was to isolate this specific message as originating from Alice, they would not be able to derive the recipient through any kind of differential attack (as all attacks would also implicate Eve).

Draft

To construct such a tag Alice runs `FlagEntangled`($\{\mathsf{pk}_{\text{carol}}, \mathsf{pk}_{\text{eve}}\}$).

Alice could choose to entangle all of her messages to Carol in this way, fully implicating Eve in her message sending regardless of Eve's false positive rate. If Eve attempted to decrypt the message she would not be able to and might assume that the tag was an unlikely false positive - as such too many of these messages might cause Eve to be suspicious. However, Eve might be a well known service or bot integrated with the privacy preserving application - allowing Alice cover without worrying about triggering suspicion.

Alice could also choose to entangle each message with a different random party.

While this strategy is, by itself, vulnerable to intersection attacks; it increases the number of potential relationships any adversary needs to rule out in order to derive the resulting metadata from the communication.

When combined with a large number of parties downloading all messages (or even downloading with high false positive rates) this strategy has the effect of increasing the anonymity set of the entire system.

It is worth nothing at this point that strategies can be combined, and their effects compound. When given a tag and a set of matches an adversarial server cannot distinguish between a true and false positive, an entangled deniable send or a group broadcast - or a combination!

3.4 Forging False Positives

Alice wants to send a message to Carol, but also wants to implicate Eve to the server. However Alice doesn't have enough time or computing power to generate a tag that will fully match against Eve's full γ-length key.

Instead Alice forges an entangled tag by running `FlagEntangled`($\{\mathsf{pk}_{\text{carol}}, \mathsf{pk}_{\text{eve}}\}$), how instead of checking all parts of the key at line 11 she instead only checks up to a value l that she believes is greater than or equal to the false positive rate of Eves detection key.

To the server the tag would match both to the detection key of both Carol and Eve, but when fetched Eve would assume it was a false positive (a much more likely one than in our previous example).

References

[1] Gabrielle Beck, Julia Len, Ian Miers, and Matthew Green. Fuzzy message detection. Cryptology ePrint Archive, Report 2021/089, 2021. `https://eprint.iacr.org/2021/089`.

Draft