Generating trusted sphinx packets

Aurelien Chassagne[1[0000−1111−2222−3333]], Iness Ben Guirat[1[1111−2222−3333−4444]], Jan Tobias Mülberg[1[2222−−3333−4444−5555]], and Jean Michel Dricot[1[1111−2222−3333−4444]]

Université Libre de Bruxelles {abc,lncs}@ulb.be

Abstract. abstract

Keywords: mixnet · sphinx · malicious users

1 Introduction

Mix network, or mixnet, is an overlay network of servers (called mix nodes) that routes messages anonymously from senders to receivers [2–4, 6, 8, 11–13]. Although mixnet shares similarities with Tor [7], they differ in two main key aspects: (i) the routing in Tor is circuit-based (meaning that all packets sent by a user follow the same path for the entire session) and in mixnets is packet-based (meaning that each packet follows a different path) and (ii) packets in Tor are forwarded immediately upon receipt by the Tor nodes while in mixnets packets are delayed for a certain amount of time t in order to mitigate timing analysis attacks. These two techniques ensure that mixnets are resilient against a strong adversary who observes the entire input and outputs of the network typically called a Global Passive Adversary (GPA). Based on Loopix [13], Nym Technologies is a mixnet-based system that allow users to send traffic anonymously. Nym is a paid service where services can be integrated on the Nym network to enjoy privacy on the network level against a fee. Users of these services are then allowed to use the nym mixnet by using Nym credentials (based on the Coconut credential [14]) that they can construct after getting a certificate of paying a specefic service to use the nym network.

However, nothing prevents users from cheating who might exploit a valid Nym credential to for another service they did not pay for. This is a particular difficult problem in anonymous communication networks where the mixnodes do not know the traffic type or the final service a user is communicating with by doing layered encryption where the final IP address is only know by the last node in the path using a packer format such as Sphinx [5].

In this paper we present a scheme that creates the Sphinx header in a decentralized way based on trusted third parties while ensuring they learn nothing about the destination or the path. We aim to provide the following properties:

- Cheating users: Users's traffic is only allowed to be routed if the traffic belongs to the same service provider from the credential

- even if the majority of headers issuers are colluding, they do not know whcih service provider the user is communicating with.
- the spinx headers can not be altered.
- Verifiers can verify that the headers has not been altered without revealing the service provider.
- Unlinkability between sphinx packets, the original sphinx packet that is constructed in a centralized way provide the *unlinkability* property, meaning that an adversary can not know that two packets are connected to the same user. Our scheme that decentralize the headers creation aim at providing this same property.

We highlight related work and motivation in Section **??**, then we specify and justify our system model in Section [**?**], where we also describe the considered threat model. We then present our scheme that decentralize the creation of the Sphinx headers in Section [**?**] and the evaluation of our proposed solution in Section [**?**]. Finally we conclude and discuss future work in Section [**?**]

2 Motivation and Related Work

2.1 Mixnet systems

Since Chaum's seminal work on untraceable email in 1981 [2], there has been a great amount of research re lated to mixnets, both in system design [8–10, 13] as in mechanism and schemes that enhance the anonymity and privacy of systems [1,14]. We highlight in this section the most relevant prior work in terms of mixnet's designs.

explain and say that these systems they only work for their own service Unlike Nym that is supposed to as a privacy networks for other services (similar to Tor). Ben Guirat et. al, showed in this work that blending traffic is actually useful in terms of the privacy provided by the overall network. However unlike tor, mixnets can be expensive due to cryptographic operations. and hence also nides are being paid and incentiviiwed through a their computation loads and hence they shouldnt waste traffic for later to be dropped at the last hop.

motivate why we are focusing on nym though this aplied to other mixnets

2.2 Sphinx

3 System and Threat Model

3.1 System Model

For example let's say Signal is integrated with Nym, and Signal users who want their traffic to be anonymous instead of sending traffic directly to the Signal server, traffic will be first routed through the mixnet such that an adversary who observes the signal server and/or the device of the user can not correlate the sender with signal server and evetually the final recepient. Signal (service provider) can add an option for user who want to pay and issue a certified attribute to those users. Users then encode this attributes into a credential and sends it to validators. If the proof is valid, validators return partial signatures. Once the user collects a threshold number of these signatures, they aggregate them to form a valid credential and re-randomize it to ensure unlinkability from previous interactions. The user can then present this credential to a verifier to prove their right to access a service to show that the credential meets all necessary payment and authentication conditions. To prevent double-spending, the verifier checks that the credential has not already been used by consulting the blockchain and then commits the credential's serial number to the blockchain upon acceptance. For example, a user can obtain an certification from the Signal service provider, construct a valid credential and then use it to route traffic to another service provider they didn't pay for or simply not allowed (an illegal website). Such misuse would be detected only at the final node of the mixnet preventing the user from accessing another application. However, prior mixnodes would have already wasted computational resources processing an invalid packet. This vulnerability enables Denial of Service (DoS) attack by exhausting mixnodes computational power with illegitimate packets.

Additionally, each encryption layer includes an integrity tag, which prevents tampering and improves the network's resistance against malicious mixnodes and active adversaries. This means that each packet follows a different path rather than using the same intermediate nodes during the whole communication.

To further prevent correlation, mixnet relies on fixed-size packet format such as Sphinx packet (section 4), making it difficult for external observers to link incoming and outgoing messages at any given node. In summary, mixnets provide stronger privacy guarantees than onion routing at the cost of increased latency.

3.2 Threat Model

4 Sphinx

Sphinx packets consist of a header and an encrypted payload. The header itself contains a *cryptographic element* α (e.g. g^x or an elliptic curve point), *encrypted routing information* β, and an *integrity tag* γ, as illustrated in Figure 1.

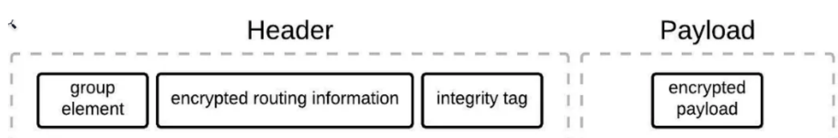

Fig. 1. Structure of sphinx packet. [source]

The *encrypted routing information (β)* is constructed in layers, applied in reverse order along the path. First, the final destination is encrypted, and an integrity tag (γ_i) is computed. The IP address of the last mixnode (n_i) is then prepended. As shown by Figure 2, this process repeats iteratively: each new header is encrypted, an integrity tag (γ_{i-1}) is computed, and the IP address of the preceding mixnode (n_{i-1}) is prepended. This layered encryption ensures that each mixnode can only decrypt its own layer, revealing the next forwarding address while preserving end-to-end confidentiality and protecting against tampering.

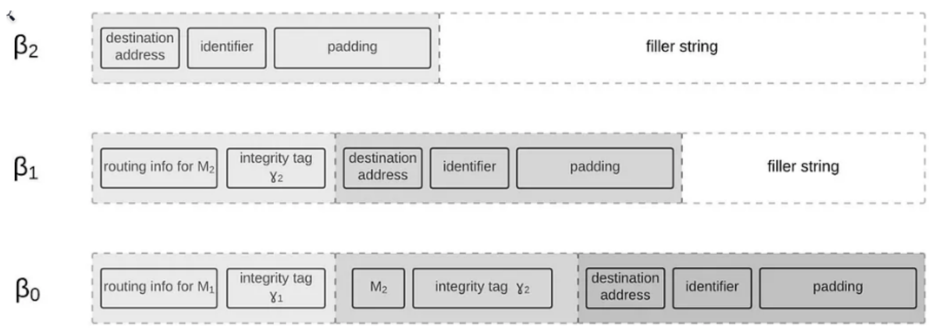

Fig. 2. Sphinx encrypted routing information encapsulation. [source]

previously "user" but JT prefers "nym client"

To encrypt the routing information, the nym client first chooses a nonce x and compute $\alpha = g^x$ as the *cryptographic element* of the header. Since each mixnode i has a private key x_i and a public key $y_i = g^{x_i}$, the user can create a shared secret s_i with mixnode i as followed: $s_i = y_i^x = (g^{x_i})^x$. Then the mixnode i receiving the packet will get α allowing him to compute the shared secret as followed: $s_i = \alpha^{x_i} = (g^x)^{x_i}$.

Instead of sending a unique *cryptographic element* α at each node in the path, the sphinx format uses a single *cryptographic element* α, which is progressively modified at each node. Each mixnode updates the cryptographic element using its shared secret as follows:

$$\alpha_{i+1} = \alpha_i^{\mathrm{hash}(\alpha_i, s_i)}$$

Thus, the user iteratively computes the shared secrets in the path's order as:

$$\begin{array}{lll} \alpha_0 = g^x, & s_0 = y_{n_0}^x, & b_0 = \mathrm{hash}(\alpha_0, s_0) \\ \alpha_1 = g^{xb_0}, & s_1 = y_{n_1}^{xb_0}, & b_1 = \mathrm{hash}(\alpha_1, s_1) \\ \vdots & \vdots & \vdots \\ \alpha_i = g^{xb_0 \cdots b_{i-1}}, & s_i = y_{n_i}^{xb_0 \cdots b_{i-1}}, & b_i = \mathrm{hash}(\alpha_i, s_i) \end{array}$$

This formulation ensures that each mixnode can independently derive the necessary cryptographic elements without requiring the full path's information, preserving privacy and unlinkability.

The *encrypted routing information (β)* is computed, as illustrated in Figure 2, by processing the path in reverse order. This involves XORing the routing information (β_{i-1}) from the previous layer (with the node's address and integrity tag) with a value derived from the shared secret s_i. Then prepending this new encrypted routing information (β_i) with an integrity tag (γ_i) and the previous mixnode address (remember we build it in reverse order). We repeat the same process for each layer (i.e. each mixnode in the path).

~\ref{}

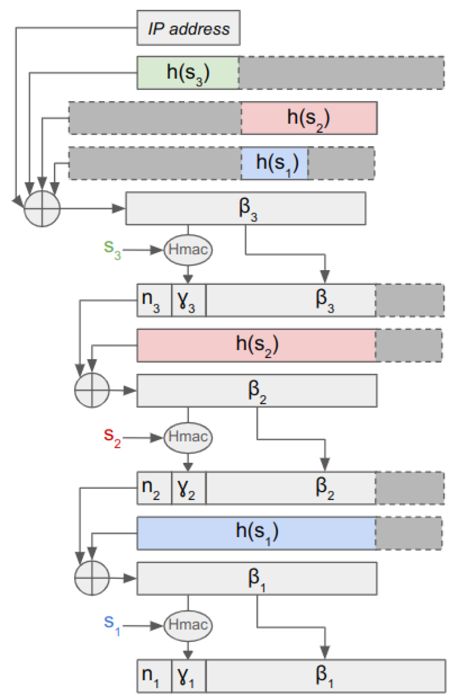

Fig. 3. Construction of the Sphinx header (modified from [5]) [TO FIX: $h(s_1)$ at first XOR]

JT: Explain what the modifications are

JT: Might help to put some (1) (2) (3) labels into the figure and refer to these labels in the text.

The first round of XOR operations differs from the others because it requires combining parts of all shared secrets. Specifically, the destination address is XORed with the last node's shared secret, truncated to match the address size. Next, the result is concatenated with the XORed values of the ending parts of the shared secrets from the other nodes in the path. This ensures that when the entire header is XORed with the full shared secret, these appended values cancel out, allowing the header to be processed in reverse order by the mixnodes. This design choice guarantees fixed-size headers, enabling fixed-size packets which is a crucial property in mixnets for maintaining unlinkability.

JT: Now maybe follow up with an example of the attack from the introduction? We should also compare computational overheads from attack vs. the new protocol design.

5 Multi-Party Computation (MPC)

The first approach to ensuring trust in the Sphinx header is to prevent user manipulation by decentralizing the header construction to Trusted Third Parties (TTP) through the use of Multi-Party Computaftion (MPC).

We consider TTPs as *honest-but-curious*. This means that they follow the protocol correctly but may attempt to infer additional information from the data they process. Our design aims to ensure that TTPs must not be able to infer any information about the shared secrets s_i nor the mixnodes involved in the path, even when TTP are colliding (assuming that at least one TTP remains honest).

What if malicous TTP...

JT: In the Nym ecosystem, who are the TTPs, who operates them, what exactly are they trusted for?

To decentralize the construction of the Sphinx header, we first examined how to partition and distribute the computation. Three approaches were considered.

The first and most naive approach involves that each TTP computes a different layer of the header. This approach reveals two consecutive nodes in the path and one of the corresponding shared secrets (s_i) at each TTP, leading to serious security concerns in the case of collusion.

In a second approach, the user sends to each TTP a piece of the destination address. Each TTP computes the same layer on its partial destination. The resulting partial headers of this layer are aggregated to compute the integrity tag. This process is repeated layer by layer. However, computing the integrity tag at the end of each layer requires knowledge of the shared secret s_i. If a TTP does it, it reintroduces the same problem as the first approach. If the user does it, he could potentially cheat.

User could send $h(s_i)$ to TPP such that it could computes the integrity check without getting useful information...

The third and retained approach is similar to the second one. Instead of partial computations at each layer, each TTP computes a full Sphinx header using only its assigned partial destination. These partial headers are then returned to the user, who aggregates them to produce the final Sphinx header. Although this approach requires less interaction with the TTPs, it introduces additional constraints as discussed in section .

ref section

JT: I think we should only focus on the third approach. There could be a section "Alternative Designs" where we discuss your other approaches.

The overall decentralized scheme is illustrated in Figure 4. The user begins by splitting the destination into several parts, such that their combination (e.g., via XOR) reconstructs the original address. Each part is sent to a different TTP, along with the required cryptographic material. The TTPs independently generate Sphinx headers using a modified version of the protocol. These partial headers are then returned to the user, who aggregates them to form the final Sphinx header, ready for transmission through the mixnet.

(see section ...)

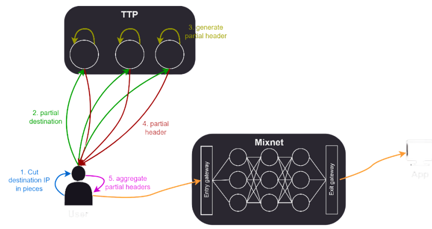

Fig. 4. Overview of the decentralized scheme

I'd rather specify the requirements for the hash here.

The main issue with the choosen approach is that we have to compute integrity tag on partial header such that combining those partial integrity tag gives the integrity tag of the final header. However, even if breaking this homomorphic hash is feasible, if it remains computationally hard enough (e.g., requiring several hours), it could still be considered sufficiently secure for our purposes.

References

1. BEN GUIRAT, I., Das, D., Diaz, C.: Blending different latency traffic with beta mixing. Proceedings on Privacy Enhancing Technologies (2023)
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981). `https://doi.org/10.1145/358549.358563`, `http://doi.acm.org/10.1145/358549.358563`
3. Chaum, D., Javani, F., Kate, A., Krasnova, A., Ruiter, J., Sherman, A.T., Das, D.: cmix: Anonymization by high-performance scalable mixing. Tech. rep., Technical report (2016), `http://www.cs.bham.ac.uk/~deruitej/papers/cmix.pdf`
4. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a Type III Anonymous Remailer Protocol. In: Proceedings of the 2003 IEEE Symposium on Security and Privacy. pp. 2–15. IEEE (May 2003)
5. Danezis, G., Goldberg, I.: Sphinx: A compact and provably secure mix format. In: 2009 30th IEEE Symposium on Security and Privacy. pp. 269–282 (2009), `http://research.microsoft.com/en-us/um/people/gdane/papers/sphinx-eprint.pdf`
6. Diaz, C., Halpin, H., Kiayias, A.: The Nym Network. `https://nymtech.net/nym-whitepaper.pdf` (February 2021)
7. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding routing information. In: Proceedings of the First International Workshop on Information Hiding. p. 137–150. Springer-Verlag, Berlin, Heidelberg (1996)
8. Hooff, J.V.D., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: Scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles. pp. 137–152 (2015), `https://people.csail.mit.edu/nickolai/papers/vandenhooff-vuvuzela.pdf`
9. Kwon, A., Lu, D., Devadas, S.: {XRD}: Scalable messaging system with cryptographic privacy. In: 17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20). pp. 759–776 (2020)
10. Lazar, D., Gilad, Y., Zeldovich, N.: Karaoke: Distributed private messaging immune to passive traffic analysis. In: 13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18). pp. 711–725 (2018)

11. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster Protocol — Version 2. IETF Internet Draft (July 2003)
12. Parekh, S.: Prospects for remailers. First Monday **1** (August 1996)
13. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). pp. 1199–1216 (2017)
14. Sonnino, A., Al-Bassam, M., Bano, S., Meiklejohn, S., Danezis, G.: Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In: 26th Annual Network and Distributed System Security Symposium, NDSS 2019. The Internet Society (February 2019)