July 2022

# White Paper // Security Principles for Private Messaging

Meta

# TABLE OF CONTENTS

# Introduction

Billions of people use Meta technologies to connect with the people who matter to them. We have a responsibility to build secure services that protect privacy and help keep people safe.

In 2019, Meta (then Facebook) announced its 'Privacy-Focused Vision for Social Networking'. To deliver this vision, we are developing our services for private messaging (the "Services") and this document sets out the principles we are following to ensure security is at the heart of our designs (the "Security Principles").

The Security Principles are the guiding values for how we are approaching securely building cross-app communication and end-to-end encryption for Messenger and Instagram Direct, and are consistent with how WhatsApp currently operates.

These Security Principles are specific to the Services, and complement our broader comprehensive security program and the range of measures we take to protect people's data. In particular, these Security Principles focus on how we think about security for private messaging, including end-to-end encryption, and how this augments our enterprise-wise practices. They should help to draw red-lines and inform difficult conversations.

# Authors

This document was produced through a collaborative effort involving engineers, security experts, and privacy teams across Meta technologies. It reflects an ongoing effort to continually review and improve company security practices, but should not be seen as a definitive or exhaustive list of the procedures and measures in place. In order to protect security and infrastructure, detailed information regarding systems and processes have been excluded from this document.

# Principles:

There are five core Security Principles, and there is welcome overlap across them.

1. **Build Secure Services for All**

These Services are built for wide scale use among those who use our technologies. We strive to provide intimate, feature-rich and user-friendly Services that provide secure messaging for billions of users, and where only the intended recipients can access end-to-end encrypted messages.

2. **Security by Design and Defense in Depth**

Private messages should be secure by design. Security should be at the forefront of how we develop the Services and be layered throughout our designs, not just an afterthought.

3. **Reduce the Attack Surface**

Minimize the opportunities for unauthorized access, including by us, to peoples' data. We work to limit the data we collect and reduce the risk of vulnerabilities by limiting complexity in our designs.

4. **Be Transparent and Invite Scrutiny**

Build transparency into our Services and where possible give people the ability to validate their security. Continue to share challenges and plans, and empower the wider security community to help critique, develop, and protect our Services.

5. **Build for the Future**

Developing secure Services is an evolution. We must build our Services with the ability to move quickly to remediate attacks (or other vulnerabilities), incorporate new technological developments, and address upcoming threats.

DETAILED GUIDANCE

# Security Design Principles

## Background

Decisions can usually be made by reference to the principles alone. However, in cases where more specific guidance is needed, we refer to the more detailed guidance below.

While it is impossible to cover every scenario we might encounter, and it is important to seek expert support whenever necessary, this detailed guidance supplements the Security Principles and helps to ensure that the Security Principles can be applied consistently and coherently in practice.

## I. Build Secure Services for All

These Services are built for wide scale use among those who use our products. We strive to provide intimate, feature-rich and user-friendly Services that provide secure messaging for billions of users, where only the intended recipients can access end-to-end encrypted messages.

*(a) Increasing Security for All*

The Services are used by billions of people across the globe and we aim to provide the highest level of security for the most people. This cannot be achieved by providing a service that is secure but not usable, for example by meeting a theoretical maximum level of security but being too difficult for many people to use correctly.

It is also important to think globally. We have people using our technologies across the world, some of whom are in low connectivity areas, using unreliable networks and infrastructure, or only have access to endpoint devices with limited functionality (including storage and processing power). It is important that our Services work in those environments as well. Again, this means providing secure services to all, rather than a theoretically maximal value of security for a select few.

*(b) Personal Trust*

It is essential that people can trust our Services. This means being transparent about our Services (to those who use our technology and the wider security community), giving people genuine control, being consistent in our messaging and ensuring that we apply our Security

Principles in practice. If we are erratic in how we approach security or deviate from our Security Principles, people (as well as the security community) will not know when they can trust us. If trust is lost, people will no longer have confidence in the privacy and security of our Services.

Trust is particularly important when designing secure Services and should be built into the product itself. Our goal is to design Services that are secure end-to-end: that data is securely transmitted, the endpoints are correct, and that content can only be made available to others by one of the recipients. If any element of this is subverted, then peoples' trust is significantly undermined. This is discussed in further detail below ('Defense in Depth' and 'Understand the System 'End-to-End'').

A way of achieving this is to couple our services with measures such as end-to-end encryption (E2EE) featuring verification and a strong authentication model. This way, people can look at their client and confirm for themselves our security protections.

*(c) Understanding People and Meeting Their Expectations*

People generally expect their messages to be secure and private. This expectation includes that only the sender and intended recipients of end-to-end encrypted messages can see, access or infer the content of those messages, and that users' messages will not be tampered with — not by hackers, criminals, governments, or even the people operating the services they're using.

When considering how we implement this expectation into the design of our Services, it is important to remember a few key points (that flow through the Security Principles more broadly):

(i) We cannot guarantee perfect, impenetrable security (nor can anyone for that matter because criminals and cybersecurity defense are both ever-evolving), but people can expect us to strive to provide strong security across all elements of the system within our control.

(ii) People don't typically pick our services because they necessarily prefer our underlying technology or the exact security settings we have implemented. Rather, they trust us to pursue security assiduously, carefully, with strict processes and layers of defense as explained in Principle II. The best security requires a real commitment to rigorously develop and maintain defenses over time.

(iii) Giving people control and the ability to validate the security of their private communications promotes trust in our Services and reduces the need for people to rely on us (and our representations).

(iv) Most people do not have an in-depth technical understanding of E2EE, or of our systems. When designing for their security, we should aim to align with their intuition of

privacy for the different things that they share, instead of expecting all users to act as idealized security experts.

To give a couple of practical applications of how this applies to designing the Services:

- E2EE is just one element in delivering the high level of security that we aim for in the Services we are building. The rollout of E2EE across our Services has attracted significant attention, but E2EE is often used as a short-hand for the broader concept of a secure end-to-end system, i.e. that the data is securely transported and, having arrived at the intended endpoint(s), cannot then be automatically leaked back to the service provider or a third party. It is this expectation (of a secure end-to-end system) that we should reflect in the design of our Services.

- We cannot expect everyone to have a detailed understanding of our security settings or the current 'state of the art' of the security landscape. We need to keep it simple for people and design our Services accordingly. This includes using clear communications; having strong default security with alerts; ensuring our designs are not too 'noisy'; pre-empting risks that people may not immediately think of and incorporating security improvements before they suffer a loss.

*(d) Appropriate Personal Choice and Control*

Choice and control has multiple layers but, at its core, we want people to be able to choose who they communicate with and have control over what they communicate.

It is therefore important to give people choice and control over how they use our Services, but we must make it easy for them to use our security tools to help protect their accounts.

Comprehension matters. This goes beyond transparency. We want people to be able to understand how our Services work and ensure that they are not inadvertently exposing themselves to security risks or unwanted contact. This means making secure approaches the easiest ones for people, including strong default security, simple and accessible controls, and a small set of consistent security options.

## II. Security by Design and Defense in Depth

Private messages should be secure by design. Security should be at the forefront of how we develop the Services and be layered throughout our designs, not just an afterthought.

*(a) Defense in Depth*

We take a "defense-in-depth" approach to security, meaning we layer a number of protections designed to prevent and address vulnerabilities in our Services from multiple angles.

No systems can provide absolute security guarantees and it's practically impossible to write flawless software at scale, so it's not uncommon to have bugs. Therefore, it is important that we incorporate multiple layers of controls to provide confidentiality and integrity.

When developing our Services, we should always keep this in mind: how can we layer our security protections to minimize risk even when any individual control may fail? For example, E2EE and reducing server-side data collection are key parts of defense in depth. These help to ensure that even if our systems are compromised, messaging content is still protected and minimal data is available to potential criminals.

Our Services should also be designed to prevent a malicious person from silently adding devices to an account. This would include integrating device-side permissions or security alerts into our designs. These designs would allow people to confirm that we are delivering messages to every listed device and, perhaps more importantly, verify that their communications are not being leaked. In summary, the design of our Services (including associated security alerts) must give people control over the endpoints of their communications and the ability to verify the maximum set of devices that can receive their messages. These processes also act as a deterrent to criminals or other malicious parties, who are less likely to target our Services (and people on our platforms) if they will be detected.

Defense in depth also helps ensure our services are built for resiliency and mitigates against the risk of service disruptions.

*(b) Understand the System 'End-to-End'*

It is important to understand our Services 'end-to-end.' This means taking into account each layer of the Service and, in particular, points under our control at which data could be stored, manipulated, or rendered. Your security is only as strong as your weakest link.

For example, discussion often focuses on encryption, however, the principal cryptographic algorithms and protocols that we rely on are tried and tested. Some of the bigger security challenges are likely to arise in the other layers. There is little benefit in having a solid lock on the front door if the ground floor window won't shut. We must apply the Security Principles across all layers of our messaging stack.

Another important consideration is endpoint security and associated security alerts. We don't want to securely deliver private communications only for people to face external compromises (see 'Defense in depth' above).

*(c) Secure-by-Default Frameworks*

We cannot expect all engineers to be experts in all aspects of security and privacy. When designing our Services, we should build and use frameworks that help ensure our engineers build technology that is secure from the very beginning.

Frameworks are development building blocks, such as secure programming languages and libraries of common bits of code, that provide engineers with built-in safeguards as they write code. These are designed to ensure that the easiest way for our engineers to achieve their objectives is to develop their code in accordance with a framework that has security and privacy controls built-in. These frameworks also make it harder to adopt 'unsafe' approaches that might inadvertently undermine security and privacy.

By developing these frameworks, we make strong security as easy and reliable as possible to achieve.

*(d) Software is Developed in a Secure Way*

Since we can't anticipate and prevent all issues when designing our Services and developing our code, it is important that we also subject the Services to robust testing. This includes the use of sophisticated analysis tools.

There are many different types of vulnerability detection tools we can deploy, including static analysis tools, which review written source code, and dynamic analysis tools, which run the code to observe errors as the program is running. These tools look for potential issues so they can either be fixed or flagged for further analysis. This enables us to find security errors at scale and as quickly as possible.

We also have world-leading engineers and security experts working at Meta; we should continue to leverage this resource and internal scrutiny is welcomed when developing the Services. Changes should not be hidden from internal review.

Certain new features of our Services will also undergo design reviews, in which security experts provide feedback to help engineers spot any deficiencies that could lead to security or privacy problems. These internal reviews provide another layer of scrutiny to help ensure we are following industry best practices.

To imagine how our products could be attacked, we also run threat modeling exercises, in which we try to anticipate how malicious people could target our Services.

No matter how we find a security bug, we respond by triaging the issue and then doing a root-cause analysis, which allows us to learn from each bug to prevent it — or errors like it — from occurring in the future. This analysis then feeds back into the other phases of our defense-in-depth approach. For example, it may lead to us building new coding frameworks, new tools, or new training.

## III. Reduce the Attack Surface

We want to minimize the opportunities for unauthorized parties, including us, to access peoples' data. We work to limit the data we collect and reduce the risk of vulnerabilities by limiting complexity in our designs.

*(a) Reducing Attack Surface*

Reducing the attack surface is inherently part of security by design, but we think it is sufficiently important to merit its own principle.

The attack surface is all of the points which might be accessible to a third party, including an attacker. It can also be difficult to identify attackers, so the most secure option is to treat all third party access as a potential vector of attack. Therefore, when designing a Service, we should only expose the surfaces necessary for its operation. If would-be attackers can't reach a surface, it makes it harder to attack it.

This is one of the reasons why we often refer to the server as a security adversary when designing our Services. Treating the server as a security adversary includes limiting its access to data and adopting client-centric architecture that puts application logic on the person's device. If the server is not part of the trust model, it significantly reduces (or even eliminates) the technical ability for a third party to gain access to messages from the provider. For example, ensuring encryption key management takes place client-side would prevent unauthorized parties from being able to access messaging data if a server is compromised.

Further, by designing our systems to prevent unnecessary access to messaging content, we also change the economics of a compromise. We are simply a less valuable target to criminals and others who plan to do harm, who are significantly less likely to invest resources seeking to exploit our systems with little prospect of reward.

In summary, adopting a client-centric architecture puts the application logic on the device, significantly reducing the risk of data exposure.

*(b) Limiting Our Access to Data and the Data We Collect*

This is an important aspect of reducing the attack surface since data that is accessible to Meta is in a position to be targeted and stolen by people looking to cause harm

We have sophisticated systems for detecting internal abuse, but this risk cannot be entirely mitigated. A major advantage of E2EE is that it reduces the attack surface on message content even in cases of insider attack. It also reduces the economics of an attack, making us a less appealing target (see 'Reducing attack surface' above).

Where we do need to collect data in order to deliver the Services, we should design our Services to do so with as coarse granularity as possible. This ensures we process data in the least

intrusive way and minimize the utility of the information to criminals and others who aim to cause harm. For example, we should consider whether we can anonymize, de-identify, or pseudonymize data to minimize our access to personal data/personally identifiable information, aggregate data, sample data; or, if we can, even add statistical noise. We should also design our Services to limit how this data can be used and only retain it for as long as is necessary, for example ensuring we delete data when it is no longer required for the operation of the Services.

We should also consider how we can minimize ways in which we may inadvertently be exposed to content, for example through certain logging functions or crash reports.

*(c) Reducing Complexity*

Additional code can make products harder to maintain, add complexity, and risks regressions, bugs and unforeseen vulnerabilities. While not every line of code definitively adds a vulnerability, it could increase the chances of one occurring, growing the attack surface.

When designing our Services, we should therefore remove any unnecessary functionality, especially where unauthorized use would be damaging to our Services and risk compromising security. Remember, if functionality exists for *authorized* people, then it can be abused by *unauthorized* people in the event of a compromise.

We should also try to reduce complexity for our user interface. Simple and easy-to-use Services increase trust and make it easier for people to choose how to use our products in a secure way (see ‘Personal trust’ above).

Reducing complexity is also important for the engineers working on our products. If we can minimize the instances where the majority of our engineers need to engage with the more complex areas of our core product designs, this reduces the risk of inadvertent compromises when developing our Services. For example, when it comes to cryptography, designing new techniques is incredibly difficult to do well, but we can leverage tried and tested algorithms and protocols. This way, the vast majority of our engineers will not need to consider the E2EE layer when designing our Services.

The secure frameworks we have in place (see ‘Secure-by-default frameworks’ above) will also help remove complexity when developing the Services.

## IV. Be Transparent and Invite Scrutiny

Our aim is to build transparency into our Services and where possible give people the ability to validate their security. Continue to share challenges and plans, and empower the wider security community to help critique, develop and protect our Services.

*(a) Transparency to People Using our Technologies, the Public, and the Global Security Community*

It is important that we are open about our Services and how we approach security. As a litmus test, when developing our Services, we should be willing to talk publicly about how we have designed security into our products. It is critical that people who use our technologies and the global security community understand how our products work and the security afforded.

To build peoples' trust, it is essential that we are able to explain how our Services work in a clear and coherent way and build transparency into our Services, including device-side controls and verification and authentication processes (see 'Building for all'). Our Services must then deliver in a consistent manner, with no nasty surprises or unexpected exceptions. People without a comprehensive understanding of the details should still be able to expect strong security and privacy for their communications.

We should also provide technical detail to the global security community. Comprehension matters; we want the global security community to be able to understand how our Services work and have sufficient technical level of detail to scrutinize our Services.

This also helps to build the trust and confidence of people who use our technologies. Not only do we give them the tools to verify their own security, but they can also rely on the scrutiny and commentary of the wider global security community. They do not just have to take our word for the security of our Services. A good example was when the security community rallied around WhatsApp, calling for responsible and contextualized reporting following an alarmist and inaccurate news article (see http://technosociology.org/?page_id=1687).

*(b) Bug Bounty Program*

Beyond being transparent with the global security community, we will also encourage outside information security experts to scrutinize our Services and provide their expertise to us through Meta's bug bounty program, one of the longest running in the industry.

We encourage security researchers to responsibly disclose potential issues so we can fix the bugs, publicly recognize their work, and pay them a bounty for finding any impactful security vulnerabilities. Our bug bounty program has been instrumental in helping us quickly detect new bugs, spot trends, and engage the best security talent outside of Meta to help us keep the platform safe. The lessons learned from each report feed back into our larger security effort, making us better and faster at finding, fixing and preventing bugs.

## V. Build for the Future

Developing secure Services is an evolution. We must build our Services with the ability to move quickly to remediate attacks (or other vulnerabilities), incorporate new technological developments, and address upcoming threats.

*(a) Foresee Challenges and Opportunities Ahead*

This isn't the end of the journey. Security is constantly evolving, and our Services should be built to accommodate change, whether that is necessary to address a vulnerability or to incorporate new developments in security technology.

Technology is also evolving. For example, there is ongoing research about improvements to authentication security and how to minimize the amount of data which must be collected in order to transmit messages across E2EE services. We should be in a position to leverage such developments and continue to develop our Services.

Even where we follow all these Security Principles, it is not possible to foresee all potential vulnerabilities. There remains the risk of compromise, unforeseen consequences of complex interplay across code, criminals and other parties developing their tactics, and new (or unknown) types of threat. We work to continually improve our defenses so we can counter emerging threats and stay ahead of our adversaries, which means that this type of work is never finished, and neither is the design of our Services.

*(b) Have the Ability to Move Quickly (and Securely)*

As well as identifying potential improvements to our Services, we also need to be able to implement them expeditiously and securely.

This supports reducing complexity in our Services (see 'Reducing Complexity' above). We should assume that engineers building new features, working on future compromises, or building additional security measures into our Services do not understand the nuances of the design decisions taken when building the Services — nor should it be necessary. By applying these Security Principles, they should be able to develop the Services to address future challenges and take advantage of future opportunities.

From a practical perspective, we should also consider how the different layers of our end-to-end system interoperate. For example, if there is independence across layers, it could make it easier to work on improvements and address vulnerabilities without impacting the other layers, making prospective changes easier to implement and reduce the risk of unforeseen complications in other layers.